

Radio Shack®

TRS-80™ Model II Disk Operating System Reference Manual

*A Description of the Operating System:
General Information, Operator Commands,
Technical Information*



CUSTOM MANUFACTURED IN THE U.S.A. BY RADIO SHACK



A DIVISION OF TANDY CORPORATION

TRSDOS

Contents

0. New Release Update	0/1
General Instructions	0/2
TRSDOS Improvements and Additions	0/7
BASIC Improvements	0/10
1. General Information	1/1
Introduction	1/3
Memory Requirements	1/5
Loading TRSDOS	1/6
Using the Keyboard	1/7
Entering a Command	1/9
File Specification	1/12
2. Library Commands	2/1
3. Utility Programs	3/1
BACKUP	3/4
FORMAT	3/6
BASCOM	3/9
COMSUB	3/9
DOCOM	3/9
DATM	3/12
EXDATM	3/12
DATM Source Listing	3/14
HERZ50	3/27
PATCH	3/28
TERMINAL	3/33
XFERSYS	3/55
4. Technical Information	4/1
Diskette Organization	4/3
Disk Files	4/4
How to Use the Supervisor Calls	4/9
List of Error Codes and Messages	4/10
Supervisor Calls	4/13
Programming with TRSDOS	4/83
5. Appendix	
TRSDOS Character Codes	5/3
Keyboard Code Map	5/7
Decimal-Hexadecimal Conversion	5/11
Numerical List of TRSDOS SVC's	5/13
6. Index	

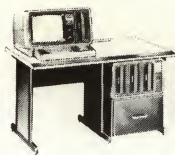
Section 0

New Release Update TRSDOS 1.2

This section pertains to the latest release of TRSDOS. Of special importance are the steps given for preparing (formatting) data diskettes, and for making duplicate (backup) copies of the TRSDOS diskette.

Before attempting to follow the steps given in this section, be sure you have read Section 1 of this manual.

Note to New Customers: *Ignore references to TRSDOS 1.1; these are for customers who are upgrading from 1.1 to 1.2 In particular, ignore Steps Three and Seven and the paragraphs describing improvements to TRSDOS and BASIC.*



General Instructions

In some of the steps below, the procedures differ for one-drive and multi-drive customers.

Step One. Use TRSDOS 1.2 FORMAT

TRSDOS 1.2 has an improved FORMAT utility. Diskettes formatted under the 1.2 utility will allow significantly faster access.

Note: Diskettes formatted under 1.2 can still be used by TRSDOS 1.1, and diskettes formatted under TRSDOS 1.1 can still be used by TRSDOS 1.2

One-drive customers:

Start TRSDOS 1.2 and format a few diskettes, using a command like this:

```
FORMAT :Ø { ID=TRSDOS, PW=PASSWORD, FULL }
```

Multi-drive customers:

Start TRSDOS 1.2 and format a few diskettes, using a command like this:

```
FORMAT :1 ID={TRSDOS, PW=PASSWORD, FULL }
```

For diskettes that will contain TRSDOS, we suggest you use ID=TRSDOS and PW=PASSWORD. For your data diskettes, use any diskette name (ID=) and any password (PASSWORD=).

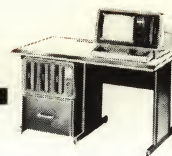
Step Two. Make a backup copy of TRSDOS 1.2

All customers:

Backup the factory-release copy of TRSDOS 1.2 onto one of your newly formatted diskettes, using a command like this:

```
BACKUP
```

Note: The password for TRSDOS 1.2 is PASSWORD.



Step Three. Upgrade your 1.1 diskettes

One-drive customers:

- 3-A. Reset the system using the old version of TRSDOS (1.1).
- 3-B. Use TRSDOS 1.1 BACKUP to duplicate the old diskette onto one of your newly formatted diskettes (from Step 1).
- 3-C. Reset the system using the new version of TRSDOS (1.2).
- 3-D. TRSDOS 2.1 contains a special utility for upgrading an old system diskette to a new system diskette. The utility is called XFERSYS. Use it to upgrade the diskette you created in Step 3-B (TRSDOS 1.1 on a new-format diskette). Under TRSDOS 1.2, type:

XFERSYS

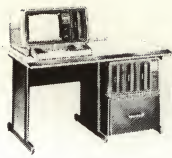
The utility will start and prompt you to swap the old ("user") and new ("master") diskettes repeatedly, until the process is completed. For details, see XFERSYS, page 3/55.

Note: The factory-release copy of TRSDOS 1.2 may not have enough room to accommodate certain Radio Shack applications programs. Create a "minimal" system diskette (Steps 4 and 5) and use this as your "master" for XFERSYS. Your "minimal system" **must** include XFERSYS or you can't use it as a "master."

Multi-drive customers:

Use only diskettes created according to Step 2 in drive zero. If you want to put some of your own files on this diskette, use COPY to do it. You do not *need* to use XFERSYS (you *may* use it, though).

You can continue using your existing data diskettes in drives 1, 2 and 3; however, you will get faster I/O from these diskettes if you back them up onto diskettes formatted by TRSDOS 1.2 (See Step 1).



MODEL II TRSDOS

Step Four (Optional). Purge unnecessary system files

Make a backup copy of the TRSDOS 1.2 diskette. Keep your factory-release diskette in a safe place, and use the backup copy.

To allow one diskette to serve both 32K and 64K RAM customers, certain files exist in two forms, one for each memory configuration. TRSDOS senses how much memory is in your system, and uses the appropriate files. The others are unused.

If you wish to free up space on your diskette, you may delete those files that are not used by your system. However, this step is optional, and should be taken carefully, so you don't delete the wrong files.

Use the TRSDOS PURGE command to delete the unnecessary files. To PURGE a diskette, you must know the password. The password on your TRSDOS diskette is PASSWORD. To start the purge, start TRSDOS (TRSDOS READY mode) and type:

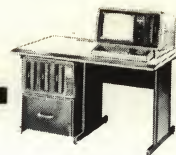
```
PURGE :Ø ALL
```

Customers with 32K RAM systems may purge only the following files (Be sure you have backup copies of them first!):

File name	Frees this many granules
SYS15B/SYS	3
SYSCB/SYS	1
SYSDB/SYS	1
DATM64	1
EXDAT64	1
BASCOM64	1
COMSUB64	1
DOCOM64	1

Customers with 64K RAM systems may purge only the following files (Be sure you have backup copies of them first!):

File name	Frees this many granules
SYS15A/SYS	3
SYSCA/SYS	1
SYSDA/SYS	1
DATM32	1
EXDAT32	1
BASCOM32	1
COMSUB32	1
DOCOM32	1



Step Five (Optional). "Minimal System Diskette"

If you want to free up still more space for your own files, you can create a "minimal system" diskette. We do not recommend this, except for special purposes. For example, certain applications programs require a great deal of space for data storage. You might want a minimal system on such diskettes. For such purposes, you may purge only the following files, in addition to those mentioned in step 2.

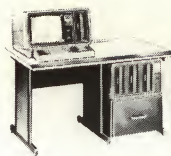
Customers with 32K RAM systems may purge only these additional files (Be sure you have backup copies of them first!):

File name	Frees this many granules
DATM32	1
EXDAT32	1
BASCOM32	1
COMSUB32	1
DOCOM32	1
TERMINAL	3
PATCH	4
XFERSYS*	3
HERZ50	1

Customers with 64K RAM systems may purge only these additional files (Be sure you have backup copies of them first!):

File name	Frees this many granules
DATM64	1
EXDAT64	1
BASCOM64	1
COMSUB64	1
DOCOM64	1
TERMINAL	3
PATCH	4
XFERSYS*	3
HERZ50	1

*If you purge XFERSYS, you can't use that diskette as an XFERSYS "master."



MODEL II TRSDOS

Step Six (Optional). Default date and time

To disable the DATE and TIME questions that come up during initialization, type in the following command:

```
PATCH TRSDOS/SYS A=1670, F=CDBB17, C=000000
```

During subsequent start-ups, TRSDOS will skip the DATE and TIME questions, setting the date to 00/00/00 and the time to 00.00.00.

By adding an AUTO command input, you can then make the system start running a program without any operator action beyond turning on the computer and inserting the diskette.

To restore the DATE and TIME questions to the start-up procedure, type in the following command:

```
PATCH TRSDOS/SYS A=1670, F=000000, C=CDBB17
```

Note: For details on the PATCH utility, see the new pages for PATCH.

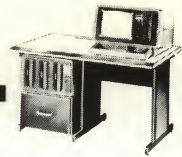
Step Seven. Convert BASIC programs to record length 256

BASIC 1.2 saves non-ASCII programs using a record length of 256 instead of 1. This allows faster loading and saving. To take advantage of this feature with your old BASIC programs, use BASIC 1.2 to load and re-save all non-ASCII programs you created under BASIC 1.1 .

For example, if you have a 1.1 BASIC program called PROGRAM, then start BASIC 1.2 and execute these commands:

```
LOAD "PROGRAM"  
SAVE "PROGRAM"
```

Files saved *with* the ASCII option (SAVE *file*, A) still have a record length of 1. Therefore you do not need to load and re-save them.



TRSDOS Improvements and Additions

FORMAT Utility

As explained under General Instructions, FORMAT has been modified so that:

- It formats diskettes faster
- The resultant diskettes allow faster I/O than do those created under Version 1.1.

Furthermore, the QUICK and FULL options are now effectively the same. Use either one to get a full diskette checkout.

DATE Command and DATE Routine

Dates are now accurate for years past 2199.

COPY command

Now allows single-drive copy to multiple diskettes even when both diskettes have the same name. For example, now one-drive customers can copy files from one TRSDOS diskette onto another, even if both diskettes have the same diskette name.

Important note: For such a copy, both diskettes must have the same version number, e.g., 1.2.

SETCOM command

For a baud rate of 110, supply 110 (**not** 100) for the baud rate parameter.

New video display codes (scroll mode only)

Decimal Code	Video Display Function
4	Steady (non-blinking) cursor (Code 1 restores blinking cursor)
30	Set 80 character/line mode and clear display
31	Set 40 character/line mode and clear display

To output these codes from BASIC, use PRINT CHR\$(code), for example:

```
PRINT CHR$(31)
```

sets the 40 character/line mode and clears the display.



MODEL II TRSDOS

New uniform set of prompt messages

For example:

ENTER COMMAND LINE (1-80)

tells you to type in a command line from one to 80 characters.

READY? (Y/Q) . .

means type Y **ENTER** when ready to continue with the operation, or type Q **ENTER** to quit the operation.

DO command

The DO command now shuts off immediately before the last command line in the file is executed. This means that you can end a DO-file by starting BASIC or some other program. When you do this, pressing **BREAK** will not automatically return you to TRSDOS. This improvement also means you can include a DO command as the last line in a DO-file; i.e., you can chain DO-files.

For example, if the last line in a DO-file (created with BUILD) is:

BASIC MENU

then after all previous lines are executed by DO, TRSDOS will start BASIC and run the program MENU. The **BREAK** key will function normally.

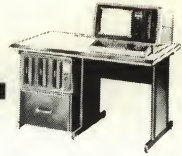
See the new page for DO.

DEBUG Upload Function

Upload no longer automatically goes to channel B at a pre-determined baud rate, word length, etc. Before using Upload, use SETCOM to initialize one of the channels — however you want it. When you start the Upload function, DEBUG will ask you which channel to use, A or B. Select the one that you initialized previously.

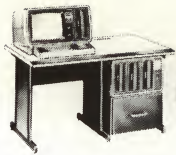
Serial Printer Driver Option

In addition to the parallel interface printer driver, TRSDOS now has a serial interface driver which you can select via the FORMS command. (See the replacement page for FORMS.)



Five new utility programs

- **TERMINAL** A full-featured terminal program to replace the TERM program of version 1.1
- **XFERSYS** A program to upgrade your TRSDOS 1.1 diskettes to TRSDOS 1.2 diskettes without affecting the user files on the diskette. This utility will be of primary benefit to one-drive customers.
- **PATCH** A program that lets you make minor changes directly in a program or data file — without having to recreate the entire file. This utility will also let you incorporate any Radio Shack-supported modifications to TRSDOS.
- **DATM** A subroutine that performs frequently used date calculations. This is supplied in two versions, one for 32K systems, one for 64K systems. A sample console program (EXDATM) that calls DATM is also provided, once again in two versions.
- **HERZ50** A DO-file that modifies TRSDOS so that it runs on a 50 Hz power supply. Only the video display driver is affected. This file should only be used in Europe and Australia (and other countries where the AC power supply is 50 Hz). Customers in U.S.A. can kill this file.



BASIC Improvements

BASIC Program Files

The SAVE command (without the A option) now creates files with a record length of 256, not 1 as in BASIC 1.1. This will allow faster saving and loading of programs, and faster copies using the TRSDOS COPY command. SAVE with the A option still creates files with a record length of 1.

Note: BASIC 1.2 can load BASIC 1.1's program files, but not vice-versa.

BACKSPACE

This key now functions in the edit mode; it moves the cursor back one position, without changing the current line. In any of the insert subcommands (I, X, H), it does delete the character from the line.

New video control codes

See item 3 of Improvements and Additions to TRSDOS.

GET and PUT with default record number

The first time you use a GET or PUT after opening a file, you must specify the record number. For subsequent GET or PUT statements, you can omit the record number, in which BASIC will use the record following the last record processed. In short:

For the first direct access of a file, use:

GET *buffer-number, record-number*
or

PUT *buffer-number, record-number*

For subsequent accesses, you may use:

GET *buffer-number*
or

PUT *buffer-number*

to get or put the next record.

RENUM

This command no longer adds filler blanks to line references shorter than five digits.

Section 1

General Information

TRS-80^{T.M.} Model II

**Disk Operating System
Reference Manual**

TRSDOS Version 1.2

Radio Shack[®]



A DIVISION OF TANDY CORPORATION

One Tandy Center
Fort Worth, Texas 76102

First Edition – 1979

All rights reserved. Reproduction or use, without express permission, of editorial or pictorial content, in any manner, is prohibited. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

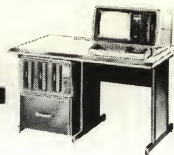
© Copyright 1979, Radio Shack
A Division of Tandy Corporation
Fort Worth, Texas 76102, U.S.A.

10 9 8 7 6 5 4 3 2

Software Copyright Notice

Model II TRSDOS is © Copyrighted 1979 by Radio Shack. All rights reserved.

Printed in the United States of America



Introduction

Model II TRSDOS (“Triss-Doss”) is a powerful and easy-to-use Disk Operating System, providing a full set of library commands and utility programs. In addition, many of the most useful System routines can be called directly by user programs.

Library commands are typed in from the TRSDOS command level to accomplish a variety of operations, including:

- Initialization—setting Printer parameters, date and time, etc.
- File-handling—copying, renaming, deleting, protecting, etc.
- File access—loading into memory, listing to Printer or Display, etc.
- Error identification

See the **Commands** section for details.

Utility programs provide essential services like:

- Formatting blank diskettes.
- Making backup copies of entire diskettes.

See the **Utilities** section for details.

System routines are executed via function codes instead of calls to absolute memory addresses. Routines available fall into seven categories:

- System control
- Keyboard input
- Video Display input/output
- Line Printer output
- File access
- Computational functions
- Serial communications

See the **Technical Information** section for details.



Notation

For clarity and brevity, we use some special notation and type styles in this book.

CAPITALS and punctuation

Indicate material which must be entered exactly as it appears. (The only punctuation symbols not entered are ellipses, explained below.) For example, in the line:

FORMAT :Ø { ID=TRSDOS, PW=PASSWORD, FULL }
every letter and character should be typed exactly as indicated.

lowercase italics

Represent words, letters, characters or values you supply from a set of acceptable values for a particular command. For example, the line:

LIST *filespec*
indicates that you can supply any valid file specification (defined later) after LIST.

... (ellipsis)

Indicates that preceding items can be repeated. For example:

ATTRIB *filespec* { *option* , ... }
indicates that several options may be repeated inside the braces.

Ø

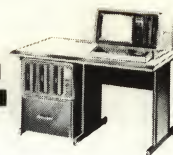
This special symbol is used occasionally to indicate a blank-space character (ASCII code 32 decimal, 20 hexadecimal).

RENAME Ø FILE/A Ø TO Ø FILE/B

x'nnnn'

Indicates that *nnnn* is a hexadecimal number. All other numbers in the text of this book are in decimal form, unless otherwise noted. For example:

X'7000'
indicates the hexadecimal value 7000 (decimal 28672).



Memory Requirements

TRSDOS occupies 6.1 tracks on the System diskette (39,040 bytes). However, only a small portion is actually in memory at any one time. The Supervisor Program, input/output drivers, and other essentials are always in memory. Auxiliary code is loaded as needed into an "overlay area".

Memory addresses 0 through 10239 (X'0'–X'27FF') are reserved for the Operating System. Certain commands, called "high overlays", also use memory addresses up to X'2FFF' (details provided in the Commands section). User programs must be located above X'27FF'; and you may want to locate them above X'2FFF' to allow use of the high overlays without loss of your program.

DECIMAL ADDRESS		HEX ADDRESS
0	<div style="border: 1px solid black; padding: 5px; text-align: center;"> SYSTEM AREA USER AREA (SHARED WITH TRSDOS "HIGH OVERLAY COMMANDS") USER AREA UNTOUCHED** BY TRSDOS MAY BE RESERVED BY TRSDOS FOR SPECIAL PROGRAMMING Last Memory Address </div>	X'0000'
10240		X'2800'
12288		X'3000'
TOP*		TOP*
32767 or 65535		X'7FFF' or X'FFFF'

MEMORY REQUIREMENTS OF TRSDOS

Note: The term "user program" applies to any program which is not a part of TRSDOS. Therefore BASIC is a user program. For memory requirements of BASIC, see the BASIC Reference Manual.

*TOP is a memory protect address set by TRSDOS. If TRSDOS is not protecting high memory, then TOP is the same as "Last Memory Address".

Single-drive COPY from one diskette to another, BACKUP and FORMAT use **all user memory.



Model II TRSDOS

Loading TRSDOS

See the **Operation Manual** for instructions on connection, power-up and inserting the System diskette.

Note: A System diskette must be in Drive 0 (the built-in unit) whenever the Computer is in use. Whenever the Computer is turned on or reset, it will automatically load TRSDOS from Drive 0.

After the System starts up, it will prompt you to enter the date. Type in the date in MM/DD/YYYY form and press **ENTER**. For example:

07/04/1979 **ENTER**

for July 4, 1979.

Next the System will prompt you to enter the time. To **skip this question**, press **ENTER**. The time will start at 00:00:00.

To **set the time**, type in the time in HH.MM.SS 24-hour form. Periods are used instead of colons since they're easier to type in. The seconds are optional. For example:

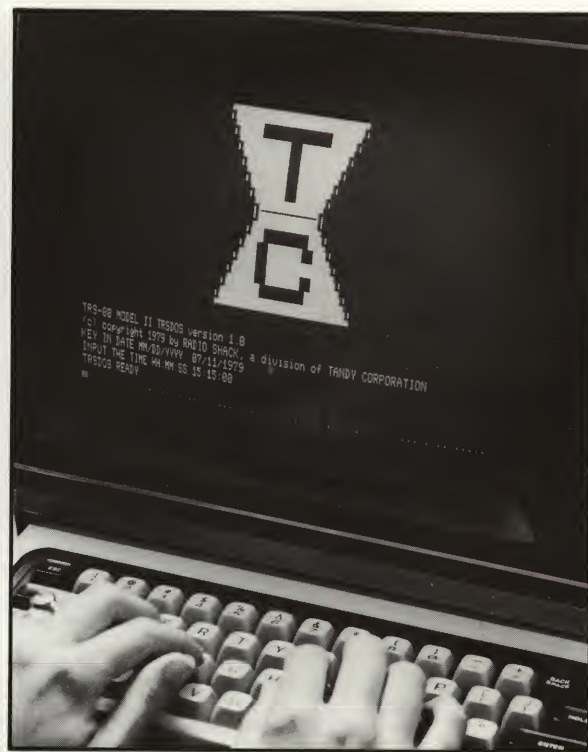
14.30 **ENTER**

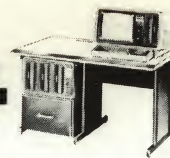
for 2:30 pm.

The System will record the time and date internally and return with the message:

TRSDOS READY

.....





Using the Keyboard

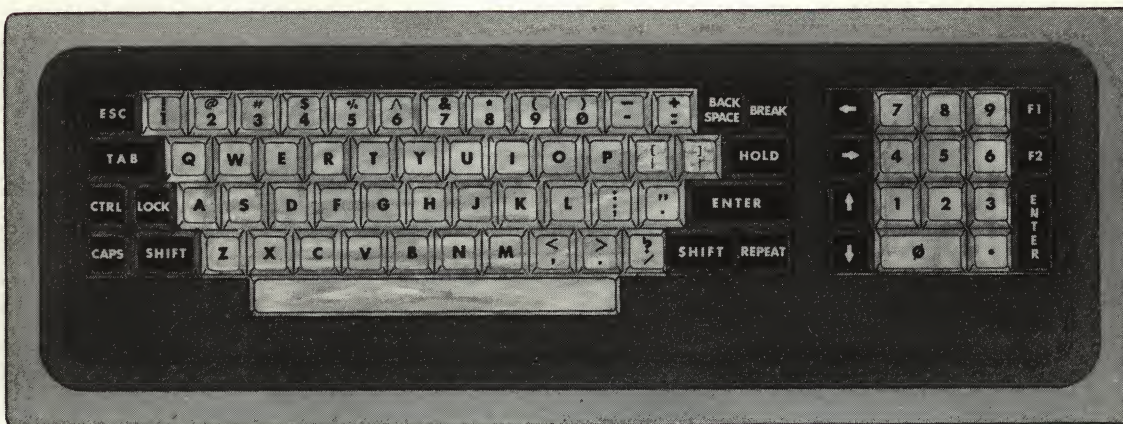
TRSDOS distinguishes between upper and lower case letters. Therefore

`dir`

is not the same as

`DIR`

Since TRSDOS commands are always capitalized, you'll probably find it convenient to operate the Keyboard in the Caps mode (press **CAPS** so the red light comes on). That way, all the alphabet-keys are interpreted as capital letters, regardless of whether the **SHIFT** key is being pressed.



Certain control keys are useful in the Command Mode:



Interrupts line entry and starts with a new line.



Backspaces the cursor without erasing any characters. Use this to position the cursor for correcting a portion of a line.



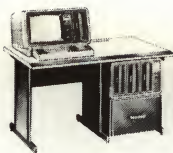
Forward-spaces the cursor without erasing any characters. Use to position the cursor for correcting a portion of a line.



Backspaces the cursor, erasing the last character you typed. Use this to correct entry errors.



Signifies end of line. When you press this key, TRSDOS will take your command. Only those characters appearing to the left of the cursor will be used.



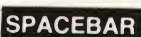
Model II TRSDOS



Pauses execution of a command. Press once more to continue.
Not functional in all commands.






Advances the cursor to the next 8-column position. Tab positions are at columns 0, 8, 16, 24, etc.

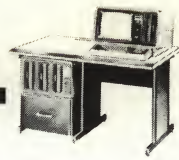


Enters a space (blank) character and moves the cursor one character forward.

If you type any other control key (non-alphanumeric, non-punctuation), a \pm symbol will be displayed for that key, but the control code will be sent to the Computer. Such control keys will either be ignored or will cause a parameter error to occur. See the Keyboard Code Map in the **Appendix** for control codes.



Repeat key. For convenience when you want to repeat a single key, hold down  while pressing the desired key. For example, to backspace halfway back to the beginning of the line, hold down  and .



Entering a Command

Whenever the TRSDOS READY prompt is displayed, you can type in a command, up to 80 characters. If the command line is less than 80 characters (as is usually true), you must press **ENTER** to signify end-of-line. TRSDOS will then “take” the command.

For example, type:

`CLS` **ENTER**

and TRSDOS will clear the Display.

Whenever you type in a line, TRSDOS follows this procedure:

First it looks to see if what you’ve typed is the name of a TRSDOS command. If it is, TRSDOS executes it immediately.

If what you typed is not a TRSDOS command, then TRSDOS will check to see if it’s the name of a program file on one of the drives.

When searching for a file, TRSDOS follows the sequence drive 0, drive 1, etc. – unless you include an explicit drive specification with the file name (described later on).

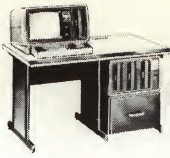
If TRSDOS finds a matching user file, it will load and execute the file. Otherwise, you’ll get an error message.

Command Syntax

Command syntax is the general form of a command, like the grammar of an English sentence. The syntax tells you how to put keywords (like `DIR`, `LIST`, and `CREATE`) together with the necessary parameters for each keyword. In this book, we present general syntax inside gray boxes, so they’re easy to recognize.

There are three general command formats:

1. No-file commands
2. One-file commands
3. Two-file commands



Syntax Forms

No-file commands

command {options} comment

options is a list of one or more parameters that may be needed by the command. Some commands have no options. The braces { } around *options* can usually be omitted when no comment is added at the end of the command line.

comment is an optional field used to document the purpose of the command-line. Comments are useful inside automatic command input files (see BUILD and DO commands).

One-file commands

command filespec {options} comment

filespec is a standard TRSDOS file specification as described later in this section.

options —See description above.

comment—See description above.

Two-file commands

command filespec-1 delimiter filespec-2 {options} comment

filespec-1 and *-2* are TRSDOS file specifications as described later in this section.

delimiter is one of the following:

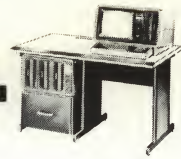
blank space or spaces (indicated as *␣*)

a comma, surrounded by optional spaces

␣TO␣ surrounded by optional spaces.

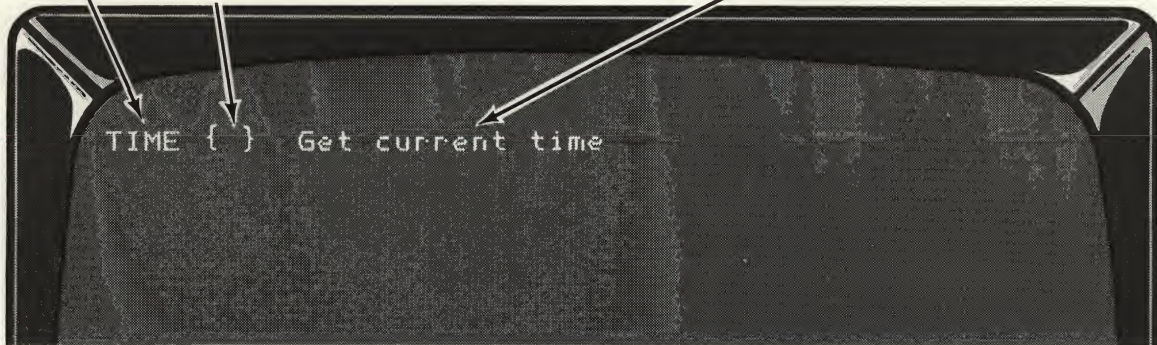
options—See description above.

comment—See description above.

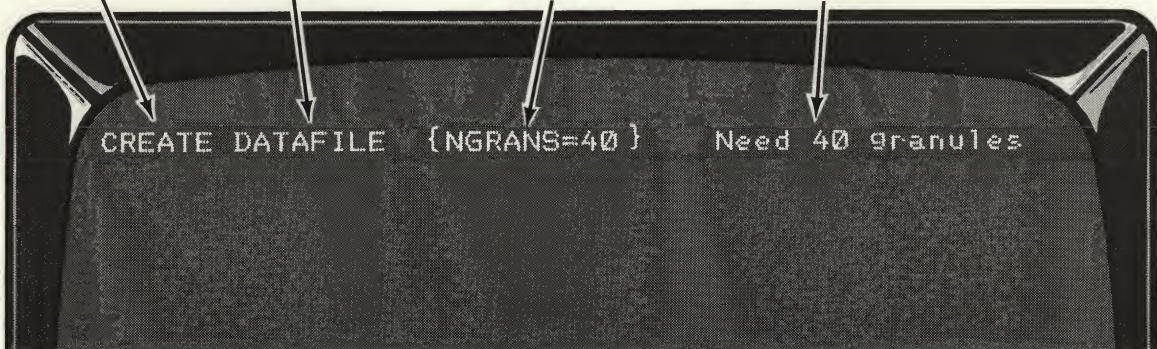


Examples of Syntax Forms

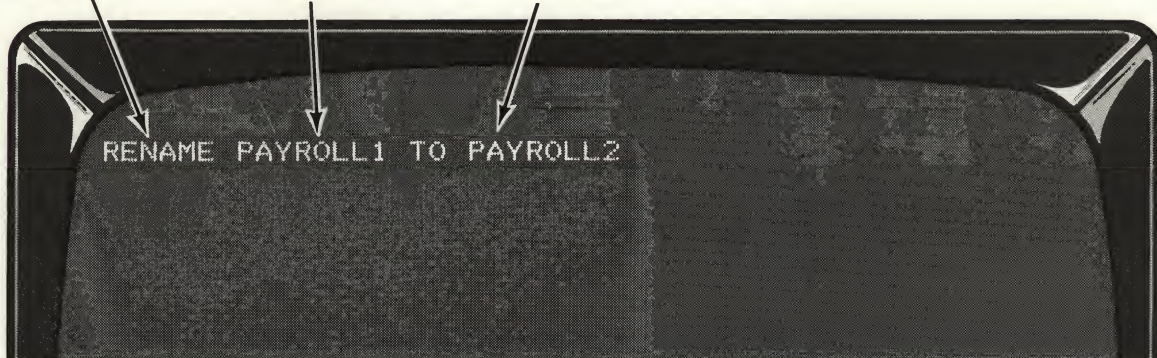
Command Empty option list
 required because of comment Comment



Command Filespec Option list Comment



Command Filespec-1 Filespec-2





File Specification

The only way to store information on disk is to put it in a disk file. Afterwards, that information can be referenced via the file name you gave to the file when you created or renamed it.

A file specification has the general form

filename/extension.password:d(diskette name)

filename consists of a letter followed by up to seven optional numbers or letters.

/extension is an optional name-extension; *extension* is a sequence of up to three numbers or letters.

.password is an optional password; *password* is a sequence of up to eight

d is an optional drive specification; *d* is one of the digits 0,1,2,3.

(diskette name) is an optional field of up to 8 letters or numbers. If this field is included, it must be preceded by a drive specification.

Note: There can be no blanks inside a file specification. TRSDOS terminates the file specification at the first blank space.

For example:

FileA/TXT.Manager:3(ACCOUNTS)

references the file named FileA/TXT(ACCOUNTS) with the password Manager, on Drive 3, diskette name ACCOUNTS.

File Names

A file name consists of a name and an optional name-extension. For the name, you can choose any letter, followed by up to seven additional numbers or letters. To use a name extension, start with a diagonal slash / and add up to three numbers or letters.

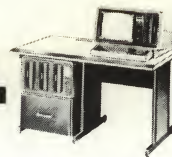
For example:

MODEL2/TXT
NAMES/123
TEST

INVENTORY
August/15
TEST1

DATA11/BAS
WAREHOUS
TEST/1

are all valid and **distinct** file names.



Although name-extensions are optional, they are useful for identifying what type of data is in the file. For example, you might want to use the following set of extensions:

/BAS	BASIC program
/TXT	ASCII text
/OBJ	Object code
/REL	Relocatable machine-language program
/DVR	Input/output driver
/SRC	Source code

Drive Specification

If you give TRSDOS a file command like:

```
KILL TEST/1
```

the System will search for the file TEST/1, starting at Drive 0 and going to the other drives in sequence 1,2,3 until it finds the file.

Anytime TRSDOS has to Open a file (e.g., to List it for you) it will follow the drive lookup sequence 0,1,2,3. When TRSDOS has to write a file, it will skip over any write-protected diskettes.

It is possible to tell the System exactly which drive you want to use, by means of the drive specification. A drive specification consists of a colon : followed by one of the digits 0,1,2, or 3, corresponding to one of the four drives.

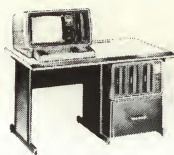
For example:

```
KILL TEST/1:3
```

tells the System to look for the file TEST/1 on drive 3 only.

Passwords

You can protect a file from unauthorized access by assigning passwords to the file. That way, a person cannot access a file simply by referring to the file name; he must also use the appropriate password for that file.



Model II TRSDOS

TRSDOS allows you to assign two passwords to a file:

- An "Update word", which grants the user total access to the information (execute, read, write, rename or delete).
- An "Access word", which grants the user limited access to the information (see ATTRIB).

When you create a file, the Update and Access words are both set equal to the password you specify. You can change them later with the PROT or ATTRIB command.

A password consists of a period . followed by 1 to 8 letters or numbers. If you do not assign a password to a file, the System uses a default password of 8 blanks. In this case the file is said to be unprotected; one can gain total access simply by referring to the file name.

For example, suppose you have a file named SECRETS/BAS. and the file has MYNAME as an update and access word. Then this command:

```
KILL SECRETS/BAS
```

will **not** cause the file to be Killed. You must include the password MYNAME in the file specification.

Suppose a file is named DOMAIN/BAS and has blank passwords. Then the command:

```
KILL DOMAIN/BAS.GUESS
```

will not be obeyed, since GUESS is the wrong password.

Diskette Names

When you reference a file like TESTER/BAS:3, TRSDOS will use whatever diskette is in drive 3. However, if you add a diskette name to the file specification, TRSDOS will first check to see that the correct diskette is in the drive. (You assign diskette names during the Format or Backup process.)

Note: Only the COPY command looks at the diskette name and checks that the correct diskette is inserted. The other commands ignore the diskette name in Version 1.2.

A diskette name consists of from 1 to 8 letters and numbers inside parentheses (). When you include the diskette name in a file specification, you must also include the drive number :d. Otherwise the diskette name will be ignored.

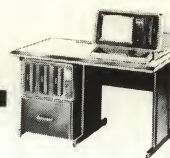
For example:

```
COPY REPORT/TXT:0 TO REPORT/TXT:3(TXTFILES)
```

tells TRSDOS to copy the file REPORT/TXT on drive 0 to another file named REPORT/TXT on a diskette named TXTFILES, using drive 3.

Section 2

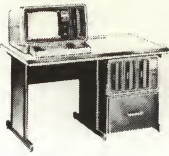
Library Commands



Introduction to Library Commands

These are the Library commands, which are typed in from the TRSDOS command level. They accomplish such functions as initializing the system, accessing or handling files, and identifying errors.

NAME	PURPOSE
AGAIN	Repeat Last Command
APPEND	Append files
ATTRIB	Change a File's Protection Attributes
AUTO	Automatic Command after System Start-Up
BUILD	Create an Automatic Command Input File
CLEAR	Clear User Memory
CLOCK	Turn on Clock-Display
CLS	Clear the Screen
COPY	Copy a File
CREATE	Create a Preallocated File
DATE	Reset or Get Today's Date
DEBUG	Start Debugger
DIR	List the Diskette Directory
DO	Begin Auto Command Input from Disk File
DUMP	Store a Machine-Language Program from RAM into a Disk File
ERROR	Display Error Message
FORMS	Set Printer Parameters
FREE	Display Disk Allocation Map
I	Initialize after Swapping Diskettes
KILL	Delete a File
LIB	Display Library Commands
LIST	List Contents of a File
LOAD	Load a Machine-Language Program File
PAUSE	Pause Execution for Operator Action
PROT	Use Diskette's Master Password
PURGE	Delete Files
RENAME	Rename a File
SETCOM	Set Up RS-232C Communications
TIME	Reset or Get the Time
VERIFY	Automatic Read After Write



Model II TRSDOS

You can enter a library command whenever the TRSDOS READY prompt is displayed. (Programs can also call library commands. See **Technical Information.**)

In general, library commands will use memory addresses below X'2800'; however, some "high overlay" commands use memory addresses up to but not including X'3000':

AUTO	DUMP	LIST	VERIFY
APPEND	TIME	PAUSE	PURGE
COPY	DATE	BUILD	
CREATE	KILL	ERROR	

See **Memory Requirements.**

General rules for entering commands

Don't type any leading blanks in front of the command. For example:

```
TRSDOS READY
  DIR
```

is an error. Omit the spaces before DIR.

There must be at least one space between the command and any option list or comment. For example:

```
DIR {1}
```

is an error. Insert a space between R and {.

There can be any number of spaces between options.

```
DIR      {SYS  ,  PRT}
```

has the same effect as:

```
DIR {SYS,PRT}
```

When no ambiguity would result, the braces around the option list can be omitted.

```
CREATE FileA NRECS=100,LRL=64
```

is acceptable, but

```
CREATE FileA NRECS=100,LRL=64  Set up file area
```

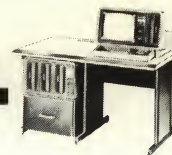
is not, since the comment "Set up file area" will be taken as an invalid parameter.

```
DIR SYS
```

is an error, since SYS is taken as an invalid drive specification. Use

```
DIR { SYS }
```

instead.



Details

When the syntax calls for a delimiter (bTOb, comma or space), other non-alphanumeric non-brace characters will also serve, unless the special punctuation is part of an option keyword, e.g., the = sign in several commands.

```
LIST TEXTFILE {PRT:SLOW}
```

is equivalent to:

```
LIST TEXTFILE {PRT, SLOW}
```




AGAIN

Repeat Last Command

AGAIN

This command tells TRSDOS to re-execute the most recently entered command.

AGAIN cannot be used after BACKUP or user program names.

Example

```
TIME  
AGAIN
```

TRSDOS will re-execute the TIME command.

Sample Use

AGAIN is useful after TRSDOS has returned an Input/Output error message instead of obeying a command. For example, suppose you type:

```
KILL OLDFILE:1
```

and the diskette in drive 1 is write protected. Then you'll get an ERROR 15 message. Put a write-enable tab on the diskette and type:

```
AGAIN
```

Now TRSDOS will re-execute the command.

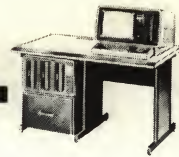
Suppose you are making multiple backup copies of a file from drive 0 to drive 1. Enter the COPY command once; for second and third copies, use AGAIN. For example:

```
COPY DAYSWORK:0 TO DAYSWORK:1
```

copies the file to a drive 1 diskette. Now put another diskette into drive 1 and type:

```
AGAIN
```

to repeat the copy using the new diskette.



APPEND

Append files

APPEND *file-1* TO *file-2*

file-1 and *file-2* are file specifications. The files must have the same type (Fixed or Variable), the same Record Length, must both be programs or both be data files (P or D in the Directory listing).

␣TO␣ is a delimiter. A comma or a space can also be used.

APPEND copies the contents of *file-1* onto the end of *file-2*. *file-1* is unaffected, while *file-2* is extended to include *file-1*. The file types (V or F) and record lengths (for fixed length record files) must match. See DIR for more information on file types and record lengths.

Examples

```
APPEND Wordfile/2 TO Wordfile/1
```

A copy of Wordfile/2 is appended to Wordfile/1.

```
APPEND REGION1/DAT,TOTAL/DAT.guess
```

A copy of REGION1/DAT is appended to TOTAL/DAT, which is protected with the password guess.

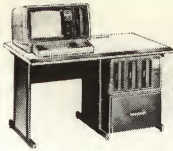
Sample Uses

Suppose you have two data files, PAYROLL/A and PAYROLL/B.

PAYROLL/A	PAYROLL/B
Atkins, W.R.	Lewis, G.E.
Baker, J.B.	Miller, L.O.
Chambers, C.P.	Peterson, B.
Dodson, M.W.	Rodriguez, F.
Kickamon, T.Y.	

You can combine the two files with the command:

```
APPEND PAYROLL/B TO PAYROLL/A
```

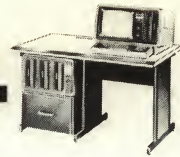


Model II TRSDOS

PAYROLL/A will now look like this:

Atkins, W.R.
Baker, J.B.
Chambers, C.P.
Dodson, M.W.
Kickamon, T.Y.
Lewis, G.E.
Miller, L.O.
Peterson, B.
Rodriguez, F.

PAYROLL/B will be unaffected.



ATTRIB

Change a File's Passwords

ATTRIB *file* {*ACC=password-1*, *UPD=password-2*, *PROT=level*}

file is a file specification.

ACC=password-1 sets the access word equal to *password-1*. If omitted, access word is unchanged.

UPD=password-2 sets the update word equal to *password-2*. If omitted, update word is unchanged.

PROT=level specifies the protection level for access. If omitted, *level* is unchanged.

Level	Degree of access granted by access word
NONE	No access
EXEC	Execute only
READ	Read and execute
WRITE	Read, execute and write
RENAME	Rename, read, execute and write
KILL	Kill, rename, read, execute and write (gives access word total access)

ATTRIB lets you change the passwords to an existing file. Passwords are initially assigned when the file is created. At that time, the update and access words are set to the same value (either the password you specified or a blank password). See Chapter 1 for details on access and update passwords.

Examples

```
ATTRIB DATAFILE ACC=JULY14, UPD=MOUSE, PROT=READ
```

Sets the access password to JULY14 and the update password to MOUSE. Use of the access word will allow only reading and executing the file.

```
ATTRIB PAYROLL/BAS.SECRET ACC=,
```

Sets the access word to blanks. The protection level assigned to the access word is left unchanged.

```
ATTRIB OLD/DAT.APPLES UPD=,
```

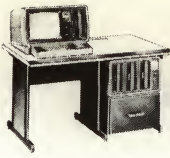
Sets the update word to blanks.

```
ATTRIB PAYROLL/BAS.PW PROT=EXEC
```

Leaves the access and update words unchanged, but changes the level of access.

```
ATTRIB DATAFILE/1.PRN PROT=,
```

Changes the access level to Kill.



Model II TRSDOS

Sample Uses

Suppose you have a data file, PAYROLL, and you want an employee to use the file in preparing paychecks. You want the employee to be able to read the file but not to change it. Then use a command like:

```
ATTRIB PAYROLL ACC=PAYDAY, UPD=Avocado, PROT=READ
```

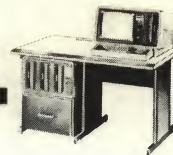
Now tell the clerk to use the password PAYDAY (which allows read only); while only you know the password, Avocado, which grants total access to the file.

Suppose you want to temporarily stop access to the file. Then use the command:

```
ATTRIB PAYROLL.Avocado PROT=NONE
```

Now the use of the password PAYDAY grants no access to the file. To restore the previous degree of access, use the command:

```
ATTRIB PAYROLL.Avocado PROT=READ
```

AUTO

Automatic Command after System Start-Up

AUTO command-line

command-line is a TRSDOS command or the name of an executable program file.

This command lets you provide a command to be executed whenever TRSDOS is started (power-up or reset). You can use it to get a desired program running without any operator action required, except typing in the date and time.

When you enter an AUTO command, TRSDOS writes *command-line* into its start-up procedure. TRSDOS does not check for valid commands; if the command line contains an error, it will be detected the next time the System is started up.

Examples

AUTO DIR {SYS}

Tells TRSDOS to write the command DIR SYS at the end of its start-up procedure. Each time the System is reset or powered up, it will automatically execute that command after you enter the date and time.

AUTO BASIC

Tells TRSDOS to load and execute BASIC each time the System is started up.

AUTO FORMS {W=80} For 8-1/2" wide paper

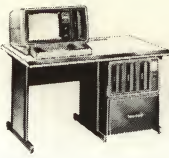
Tells TRSDOS to reset the printer width parameter each time the system is started up.

AUTO PAYROLL/CMD

Tells TRSDOS to load and execute PAYROLL/CMD (must be a machine-language program) after each System start-up.

AUTO DO STARTER

Tells TRSDOS to take automatic key-ins from the file named STARTER after each system start-up. See BUILD and DO.



Model II TRSDOS

To erase an automatic command

Type:

AUTO

This tells TRSDOS to delete any automatic key-ins and reset the start-up procedure to go directly to the TRSDOS READY mode.

Important Note

You cannot over ride an automatic command. Therefore be sure a program is fully debugged before making it an automatic command. Furthermore, programs which are executed via the AUTO function should normally provide a means of exiting to the TRSDOS READY mode. (Unless the **BREAK** key is blocked by the user program, pressing **BREAK** will get you back to TRSDOS.)

Sample Use

Suppose you want the TRSDOS to run a certain BASIC program, MENU, each time it is started up. That way, an operator can turn on the Computer and get going without having to enter any TRSDOS commands.

Then use the command:

AUTO BASIC MENU

to prepare the System to run the BASIC program each time it starts up. (See BASIC Reference Manual for details on loading BASIC.)



BUILD

Create an Automatic Command Input File

BUILD file

file is a file specification which cannot include an extension.

This command lets you create an automatic command input file which can be executed via the DO command. The file must contain data that would normally be typed in from the keyboard.

BUILD files are primarily intended for passing command lines to TRSDOS just as if they'd been typed in at the TRSDOS READY level.

BUILDing New Files

When the file you specify does not exist, BUILD creates the file and immediately prompts you to begin inserting lines. Each time you complete a line, press **ENTER**. BUILD will give you another chance to re-do the line or keep it. Press **ESC** to erase and re-do the line; **ENTER** to store it and start the next line.

While typing in a line, you can use **←** and **→** to position the cursor for corrections. **BACKSPACE** also works as usual. Be sure the cursor is at the end of the desired line before you press **ENTER**.

To end the BUILD file, simply press **ENTER** at the beginning of the line, i.e., when the message:

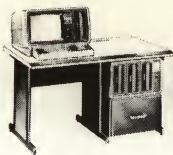
```
ENTER COMMAND LINE (1-80)
```

Is displayed.

Note: Pressing **BREAK** will also end the file. Only those lines that have been flagged like this:

```
*** LINE STORED IN FILE ***
```

will be saved.



Editing Existing BUILD-Files

When you specify an existing file in the BUILD command, TRSDOS assumes you want to edit that file. Before starting the edit, it copies the file into a new file with the same name but with the extension/OLD. That way, you will have a backup copy of the file as it was before being edited.

Note: Editing an existing BUILD-file requires that you have write-access to the file. That is, if the access password has a protection level which does not allow writing, then you must supply the update password.

Example

Suppose the file STARTER already exists, and you type the command:

```
BUILD STARTER
```

TRSDOS will first copy STARTER into a new file STARTER/OLD (if STARTER/OLD already exists, previous contents are lost). Then it will let you begin editing the file. As you edit the file, the updated lines will be written into STARTER.

BUILD will display the existing contents of the file, one line at a time. Beneath the line is an option list:

```
Keep, Delete, Replace, Insert or Quit?  
ENTER (K/D/R/I/Q) ..?
```

Type the first letter of the desired option and press **ENTER**

Keep Option: Copies the line as-is into the new file, and displays the next line for editing.

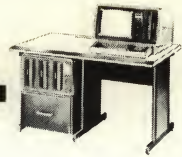
Delete Option: Deletes the line by not copying it into the new file, and displays the next line for editing.

Insert Option: Allows you to insert lines **ahead** of the line being displayed. Using this option is like entering lines into a new file as described above.

After you press **ENTER**, TRSDOS will give you a chance to erase and re-start the insert line, or to store the insert line. Press **ESC** to erase. **ENTER** to store it. You can then insert another line.

To stop inserting, press **ENTER** at the beginning of the line. TRSDOS will then display the next line and the option list.

Replace Option: Deletes the displayed line and lets you insert replacement lines. Entering replacement lines is like entering lines with the insert option. Press **ENTER** at the beginning of a line to stop inserting. TRSDOS will display the next line and the option list.



Quit Option: Ends the editing session. All remaining lines will be copied into the new file as-is. Before closing the file, TRSDOS will ask you if you want to add new lines to the end. (If you simply want to add to a file but make no other changes, type Q at the beginning of the edit session.)

At end of file

Whenever TRSDOS reaches the end of the file, it will ask if you want to add new lines at the end. Type Y **ENTER** to add, N **ENTER** to end the editing session.

Adding lines at the end of a file is just like using the insert line option described above. Type **ENTER** at the beginning of a line to stop adding and close the file.

To recover a BUILD file's previous contents

There are a couple of cases in which you may need to do this. Let's assume you are editing a file named STARTER.

1. After ending the edit session, you realize that you have made an error, and you want to recover the previous version of the file.
2. You accidentally press **BREAK** and end the edit session; only those lines that have been flagged like this:

```
*** LINE STORED IN FILE ***
```

will be saved in the new file named STARTER.

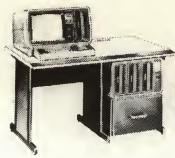
The previous file contents are now stored in STARTER/OLD. If you want to re-edit this file, you must Copy it or Rename it to a file name without an extension. For example, you might use this command:

```
COPY STARTER/OLD TO STARTER {ABS}
```

(See COPY for an explanation of the ABS parameter.) Now you can edit the previous file's contents. Type:

```
BUILD STARTER
```

to start editing.



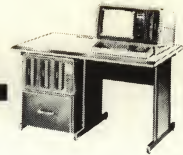
CLEAR **Clear User Memory**

CLEAR

This command gets you off to a fresh start. It zeroes user memory (loads binary zero into each memory address above X'27FF') and clears the Display. It also returns the System to the state it is in when the first TRSDOS READY message appears: initializes the input/output drivers, un-protects all memory and resets the stack.

Example

CLEAR



CLOCK

Turn on Clock-Display

CLOCK {*switch*}

switch is one of the options, ON or OFF. If *switch* is not given, ON is assumed.

This command controls the real-time clock display in the upper right corner of the Video Display. When it is on, the 24-hour time will be displayed and updated once each second, regardless of what program is executing.

TRSDOS starts up with the clock off.

Note: The real-time clock is always running, regardless of whether the clock-display is on or off.

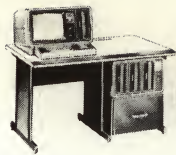
Examples

CLOCK

Turns on the clock-display.

CLOCK OFF

Turns off the clock-display.



CLS

Clear the Screen



CLS

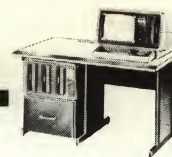
This command clears the Display. Use it to erase information that you don't want others to see, for example, file specifications which include passwords.

Example

CLS

Sample Use

```
CREATE PERSONNL/BAS.secure  NGRANS=300  
CLS
```



COPY

Copy a File

COPY *file-1* TO *file-2* {ABS }

file-1 and *-2* are file specifications.

TO is a delimiter. A comma or space can also be used.

ABS is an optional parameter telling TRSDOS to copy *file-1* even if *file-2* already exists. The previous contents of *file-2* will be lost.

This command copies *file-1* into the new file defined by *file-2*. If a disk name is included in either file specification, TRSDOS will ensure that the appropriate diskette is inserted before making the copy. This allows you to copy a file from one diskette to another, using a **single drive** if necessary.

When you do not add the ABS (“absolutely”) parameter, TRSDOS **will not overwrite** an existing file that matches the specification *file-2*. Instead it will give you an error message. Use the ABS option to overwrite (destroy) an existing file.

Normally, COPY uses memory below X’3000’; however, when copying from one diskette to another in a **single** drive, it will use memory up to the start of protected memory (see **Memory Requirements**).

The diskette name must always be preceded by a drive specification; otherwise it will be ignored.

For single-drive copies from one disk to another, both disk names must be provided.

Examples

COPY OLDFILE/BAS TO NEWFILE/BAS

Copies OLDFILE/BAS into a new file named NEWFILE/BAS. TRSDOS will search through all drives for OLDFILE/BAS, and will copy it onto the first diskette which is not write-protected.

COPY NAMEFILE/TXT:0(DEPTC) TO NAMEFILE/TXT:0(DEPTA)

This command specifies a one-drive copy from a diskette named DEPTC to another diskette named DEPTA. TRSDOS will provide the necessary prompting to accomplish the copy.



Model II TRSDOS

`COPY FILE/A TO FILE/B:1(DOUBLE)`

This command copies FILE/A to FILE/B. TRSDOS will search all drives for FILE/A, and will require you to have or insert a diskette named DOUBLE in drive 1.

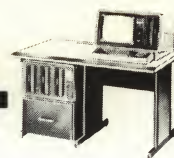
`COPY NEWFILE TO OLDFILE {ABS}`

Performs the copy even if OLDFILE already exists, in which case its previous contents are lost.

Sample Use

Whenever a file is updated, use COPY to make a backup file on another diskette. You can also use COPY to restructure a file for faster access. Be sure the destination diskette is already less segmented than the source diskette; otherwise the new file could be more segmented than the old one. (See FREE for information on file segmentation.)

To rename a file on the same diskette, use RENAME, not COPY.



CREATE

Create a Preallocated File

CREATE file {NGRANS=*n1*, NRECS=*n2*, LRL=*n3*, TYPE=*letter*}

file is a file specification.

NGRANS=*n1* indicates how many granules to allocate. If NGRANS is omitted, the number of granules allocated is determined by NRECS and LRL.

NRECS=*n2* indicates how many records to allow for. If NRECS is omitted, NGRANS determines the size of the file. When NRECS is given, LRL must also be given.

LRL=*n3* indicates the record length (Fixed-length records only). *n3* must be in the range [1,256]. If LRL is omitted, LRL=256 is used. When LRL is given, NRECS must also be given.

TYPE=*letter* specifies the record type: *letter* equals F (Fixed-length records) or V (Variable-length records). If TYPE is omitted, TYPE=F is used.

NOTE: {NGRANS} and {NRECS,LRL} are mutually exclusive.

This command lets you create a file and pre-allocate (set aside) space for its future contents. This is different from the default (normal) TRSDOS procedure, in which space is allocated to a file dynamically, i.e., as necessary **when data is written into the file.**

With preallocated files, unused space at the end of file is **not** deallocated (recovered) when the file is Closed. With dynamically allocated files, on the other hand, unused space at the end of the file IS recovered when the file is Closed.

Note: With pre-allocated files, TRSDOS will allocate extra space when you exceed the pre-allocated amount during a write operation.

You may want to use CREATE to prepare a file which will contain a known amount of data. This will usually speed up file write operations, since TRSDOS won't have to do periodic allocations during the write operations. File reading will also be faster, since pre-allocated files are less dispersed on the diskette—requiring less motion of the read/write mechanism to locate the records.

Examples

```
CREATE DATAFILE/BAS NRECS=300, LRL=256
```

Creates a file named DATAFILE/BAS, and allocates space for 300 256-byte records.

```
CREATE TEXT/1 NGRANS=100, TYPE=V
```

Creates a file named TEXT/1, and allocates 100 granules. The file will contain variable-length records.



Model II TRSDOS

```
CREATE NAMES/TXT.IRIS NRECS=500, LRL=30
```

Creates a file named NAMES/TXT protected by the password IRIS. The file will be large enough to contain 500 records, each 30 bytes long.

Determining the size of the file

You can allocate space according to number of granules or number of records. (A granule contains 1280 bytes; a record contains from 1 to 256 bytes, depending on LRL.)

The granule is the unit of allocation in TRSDOS: if you ask for 30 granules, that's exactly how much space the file will get (38,400 bytes).

If, on the other hand, you specify the number of records, TRSDOS will give you the **number of granules** which are required to **contain** that many records. For example, if you specify 100 records and a record length of 40, you're asking for a total of $100 * 40 = 4000$ bytes. Since TRSDOS allocates spaces in units of granules (1280 bytes), you'll actually get 4 granules—containing 5120 bytes.

Record Length (Fixed-Length Files Only)

A record is the quantity of data TRSDOS processes for you during disk operations. The record length can be any value from 1 to 256.

File Type

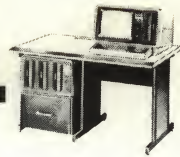
TRSDOS allows two types of files: Fixed-Length Record (FLR) files and Variable-Length Record (VLR) files. With FLR files, the record length (from 1 to 256) is set when the file is created, and it cannot be changed. With VLR files, the length of each record is independent of all other records in the file. For example, record 1 might have a length of 70; record 2, 33; record 3, 225; etc. Variable length records consist of a length byte followed by the data, and can contain up to 256 bytes **including** the length byte.

For further explanation of file structure, allocation and types, see **Technical Information**.

To Create a file to be used by BASIC

1. Decide how many records the file will contain. (This is just an estimate. If the file exceeds this number, it will automatically be extended.)
2. If it is a Direct access file, determine the optimum record length (from 1 to 256). If it is a sequential access file, the record length must equal 1.
3. Use a CREATE command like this:

```
CREATE file { NRECS=number, LRL=length }
```

Sample Use

Suppose you are going to store personnel information no more than 250 employees, and each data record will look like this:

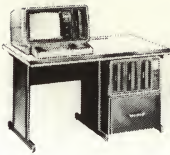
Name (Up to 25 letters)
Social Security Number (11 characters)
Job Description (Up to 92 characters)

Then your records will need to be $25 + 11 + 92 = 128$ bytes long.

You could create an appropriate file with the command:

```
CREATE PERSONNL/TXT NRECS=250, LRL=128
```

Once created, this preallocated file will allow faster writing than would a dynamically allocated file, since TRSDOS won't have to stop writing periodically to allocate more space (until you exceed the pre-allocated amount).



DATE

Reset or Get Today's Date

DATE *mm/dd/yyyy*

mm is a two-digit month specification.

dd is a two-digit day-of-month specification.

yyyy is a four-digit year specification.

If *mm/dd/yyyy* is given, TRSDOS resets the date. If *mm/dd/yyyy* is omitted, TRSDOS displays the current date and time.

This command lets you reset the date or display the date and time.

The operator sets the date initially when TRSDOS is started up. After that, TRSDOS updates the time and date automatically, using its built-in clock and calendar. You can enter any four-digit year after 1599.

When you request the date, TRSDOS displays it in the format:

```
THU JUL 19 1979 200 -- 14.15.31  
for Thursday, July 19, 1979, the 200th day of the year, 2:15:31pm.
```

Note: If the time passes 23.59.59, TRSDOS does not start over at 00.00.00. Instead, it continues with 24.00.00. However, the next time you use the **TIME** or **DATE** command, the time will be converted to its correct 24-hour value, and the date will be updated. If you let the clock run past 59.59.59, it will recycle to 00.00.00, and the date will not be updated to include the 60-hour period.

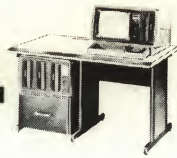
Examples

DATE

Displays the current date and time.

DATE 07/18/1979

Resets the date to July 18, 1979, and displays the new information.



DEBUG

Start Debugger

DEBUG {*switch*}

switch is one of the following parameters:

ON turns on the debugger.

OFF turns off the debugger.

If *switch* is omitted and debugger is off, TRSDOS tells you so.

If *switch* is omitted and debugger is on, TRSDOS enters the debug monitor.

This command sets up the debug monitor, which allows you to enter, test, and debug machine-language programs. It also includes an Upload function to allow transmission of data from another device to the Model II, via the built-in serial interface (Channel B).

DEBUG loads into the high memory area sometimes reserved by TRSDOS for special programming (see **TRSDOS Memory Map**). While DEBUG is on, TRSDOS will automatically protect this area from being overlaid by BASIC or other user programs. **To use DEBUG from BASIC, you must turn DEBUG on before you start BASIC.**

While DEBUG is on, every time you attempt to load and execute a user program, you will enter the debug monitor. In this mode, you can enter any of a special set of single-key commands for studying how your program is working.

DEBUG can only be used on programs in the user area (X'2800' to TOP).

Examples

DEBUG

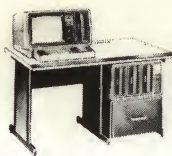
If DEBUG is off, this command tells you so. If it is on, this command enters the debug monitor.

DEBUG OFF

Turns off DEBUG and un-protects high memory.

DEBUG ON

Turns on DEBUG: i.e., loads the debugger into high memory, protects high memory, and sets up a "scroll window"—a block of lines that will be scrolled. The scroll window will consist of the bottom 11 lines of the display. The top 13 lines will be used to contain the debug monitor display.



Model II TRSDOS

To enter the debug monitor

Type:

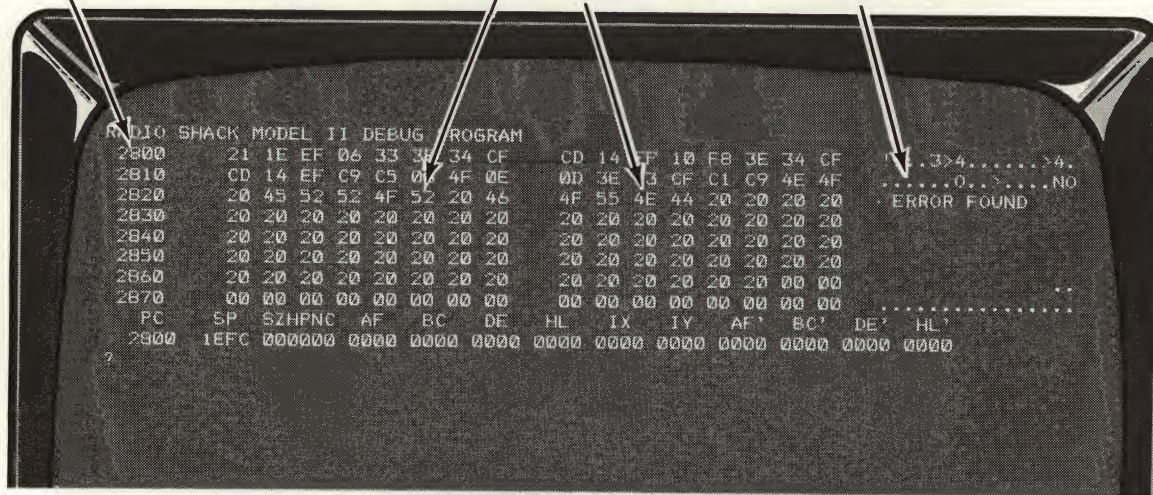
```
DEBUG ON
DEBUG
```

While DEBUG is on, you can also enter the debug monitor simply by typing the file specification of a user program. TRSDOS will load the program and transfer control to the debugger. The transfer address for the program will be in the PC register display.

Start address of one
16-byte "row" of RAM.

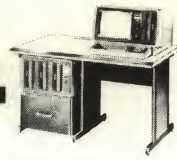
RAM display—shows
hex contents of each
byte.

ASCII display—
period "." indicates
a non-displayable
character.



Z-80A register contents.
SZHPNC are the flag bits
in register F.

The ? is the command prompt, meaning that you can enter one of the single-key commands. Press **H** (for "help") to display a "menu" or list of debugger commands. To enter one of the commands, press the letter which is capitalized in the command menu. For example, to enter the memory command ("raM"), press **M**.



Most commands will prompt you to enter additional information or subcommands. While entering commands and subcommands, the following keys are useful:

ESC	Returns to the ? prompt and cancels the command you're in.
BACK SPACE	Backspaces the cursor and erases previous character.
←	Cursor back without erasing.
→	Cursor forward without erasing.
F1	In certain subcommands, homes the cursor.
TAB	In certain subcommands, tabs the cursor.

Command Description

B (Breakpoint)

Press **B** to set a breakpoint in your program. When execution reaches a breakpoint, control returns to the debug monitor, with the program counter pointing to the breakpoint address. To continue from that point, press **C**. The original instruction will be executed—**but the breakpoint will not be removed**. It will still be there the next time that address is reached.

Note: Place breakpoints at the beginning byte of an opcode—**never** in the middle of an instruction.

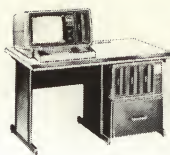
Press **B** to enter the Breakpoint command. TRSDOS prompts you to enter the breakpoint number. Up to eight breakpoints are allowed, so type in a number from 1 to 8. Next TRSDOS prompts you to enter the new address for that breakpoint. If the breakpoint has previously been set, TRSDOS displays the old breakpoint address, and the original instruction that goes in that address.

Note: While a breakpoint is in place, X'D7' is displayed in the memory display for the breakpoint address.

For example:

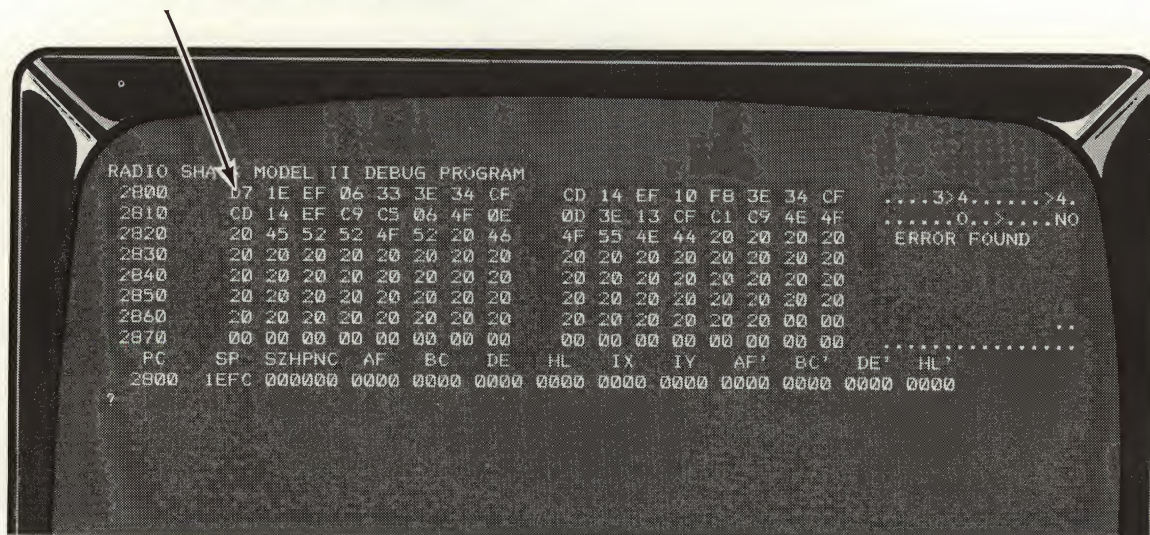
```
? B #=1 A=2800
```

Puts a breakpoint (#1) at address X'2800'. The memory display for X'2800' will show a X'D7'.



Model II TRSDOS

X'D7' indicates a breakpoint
has been set at this address.



To delete a single breakpoint without affecting any others, press **ENTER** instead of providing a new address for the breakpoint. To delete all breakpoints, press **E** for "Empty breakpoint table".

C (Continue)

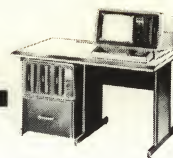
Press **C** to enter this command. It resumes execution of your program at the address pointed to by PC. Use it after the debugger has stopped at a breakpoint. The original instruction at the breakpoint address will be executed, but the breakpoint will remain in place.

D (Decimal Format)

Press **D** to enter this command. It displays all addresses in decimal form. However, the contents of all registers and memory addresses are still displayed in hexadecimal. In the decimal display format, you must enter all addresses as five-digit decimal numbers.

E (Empty Breakpoint Table)

Press **E** to empty the breakpoint table. All breakpointed instructions will be restored.



F (Find Hex String)

Press **F** to start this command. It will search in memory for a string up to 20 bytes long. You must enter the search string in hexadecimal format. Press **ENTER** when you have typed in the entire string. The debug monitor will display the first occurrence of the string. If it is not in the search area, the current memory display is unchanged.

For example:

```
F S=2800 E=4000 D=C30070
```

Searches memory from X'2800' through X'4000' for the three-byte hexadecimal string X'C30070'.

X (Hex Format)

Press **X** to restore the Display to hexadecimal format. In this mode, all addresses must be entered as four-digit hexadecimal numbers.

J (Jump)

Type **J** to enter this command. The debugger will prompt you to type in the address to jump to. (Jumping to a breakpointed instruction will cause an immediate return to the debug monitor.)

For example:

```
? J A=2800
```

starts execution at X'2800'.

L (Load or Copy memory to memory)

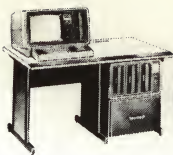
Type **L** to enter this command. It moves a block of memory. The debugger will prompt you to type in the start (S=) and end (E=) addresses of the block to be copied, and the destination address (T=) for the first byte moved.

The move is incremental: the first byte is moved to the first destination address, then the second to the second destination address, etc.

Examples:

```
? L S=2800 E=28F0 T=3000
```

Copies addresses from X'2800' to X'28F0' into memory from X'3000' to X'30F0'.



Model II TRSDOS

You can use this command to fill memory with a specific value, by putting the desired value in address *nnnn*, and using a command like this:

? L S=*nnnn* E=*xxxx* T=*nnnn*+1

This will copy the value in *nnnn* into every location from *nnnn*+1 to *xxxx*. For example, if X'2800' contains a X'20', then the command:

? L S=2800 E=3000 T=2801

fills memory from 2801 to 3000 with X'20'.

O (Debug Off)

Type **O** to exit the debug monitor and turn off DEBUG. All breakpoints set by the B command will be removed from your program, **and execution will continue at the address shown in PC. Do not use this unless you are sure there is an executable Z-80 program at (PC).** If you simply want to exit DEBUG and turn it off, type:

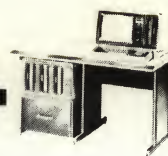
S

Then under TRSDOS READY type:

DEBUG OFF

P (Print Display)

Type **P** to send a copy of the Display to the Printer. Printer must have been initialized during TRSDOS startup or by the FORMS command.



M (Examine and Change Memory)

Type **M** to enter this command. The debugger will prompt you to type in the starting address of memory to be examined. As soon as you type in the complete address, the memory display will show the 128-byte area starting with that address. While the **A= . . .** prompt is present, you can scroll through memory 16 bytes at a time by pressing **ENTER**.

To modify any memory in the display area, press **F1** while the **A= . . .** is displayed. The cursor will move up into the memory display area.

With the cursor in the memory display area, the cursor control keys are:

← → ↑ ↓ Cursor motion: back, forward, up, down

F1 Homes Cursor

TAB Tabs Cursor

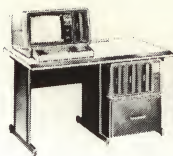
ENTER Moves cursor to start of next row

/ Moves cursor in and out of ASCII area

When the cursor is in the hexadecimal area, enter hexadecimal values. The debugger will update the memory display as you type in each nibble (hexadecimal character, half a byte).

When the cursor is in the ASCII area, enter ASCII characters. Press **/** to return to hexadecimal entry.

To cancel all changes in memory, press **ESC**. **To effect all changes**, press **F2**.



Model II TRSDOS

R (Modify Registers)

Press **R** to enter this command. The **R =>** prompt appears. Type in a letter indicating which register-pair you want to change:

A for AF	B for BC	D for DE	H for HL
X for IX	Y for IY		
F for AF'	C for BC'	E for DE'	L for HL'

The cursor will move over to the first byte of the register pair. While in the register modify mode, use the cursor control keys, **←** and **→** to move over one nibble at a time. Use **TAB** to advance to the next register pair.

To **cancel changes** in register contents, press **ESC**. To **effect changes** made, press **F2**.

S (System)

Press **S** to return to the TRSDOS READY mode. The debugger is still on; when you load and execute a program, you will enter the debugger again.

U (Upload)

Press **U** to enter the serial input mode, in which the Computer accepts serial input from another device (Model II or other computer, etc.).

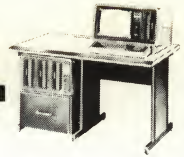
Before using Upload, you must initialize one of the serial I/O channels with the SETCOM command. Select either channel, and use the appropriate parameters for compatibility with the transmitting device. The maximum recommended baud rate is 2400.

For example, for 1200 baud, 8-bit words, no parity and 1 stop bit on channel A: press **S** to return to TRSDOS, then type:

```
SETCOM A=(1200, 8, N, 1)
```

Then type **DEBUG** to return to the debug monitor. Before starting Upload, connect the channel you selected to the transmitting device, and connect the terminator plug to the other channel if it is not already connected to another serial device.

When you are ready to start uploading, press **U**. Debug will prompt you to select a channel (C=). Type either **A** or **B**, whichever one you are going to use. Upload will then wait for the data stream to start. To exit from Upload at any time, press **ESC**.



The transmitting program must send the data in "Intel® Paper Tape Hex Format", described below.

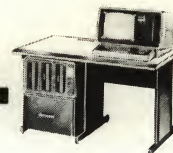
Each byte of data is sent as a pair of hexadecimal ASCII-coded characters:

- 1) high nibble (most significant four bits), sent as first byte of pair.
- 2) low nibble (least significant four bits), sent as second byte of pair.

For example, the value X'F7' is sent as two bytes, "F" (X'46') followed by "7" (X'37').

Because only ":" and ASCII coded hexadecimal numbers are sent, data is always in the range [X'30', X'3A'] or [X'41', X'46']. Values outside this range will terminate reception and produce an error message.

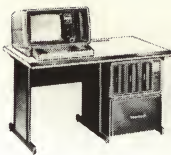
*THIS PAGE LEFT BLANK
INTENTIONALLY*



RECORD FORMAT

Records must be sent as follows:

CHARACTER NUMBER	CONTENTS	COMMENTS
1	“.”	Sync-character to indicate beginning of record.
2	High nibble of record length (N)	This 2-byte sequence gives the number of byte PAIRS in this record. Zero means 256 byte pairs follow.
3	Low nibble of record length (N)	
4	High nibble of msb of load addr.	This 4-byte sequence gives address where the data is to start loading. Address specified must be in the user area [X'2800', TOP].
5	Low nibble of msb of load addr.	
6	High nibble of lsb of load addr.	
7	Low nibble of lsb of load addr.	
8	High nibble of lsb of EOF (end of file) code	This byte-pair gives the EOF code. Any non-zero value means end of file (no more records follow). A value of zero means more records follow.
9	Low nibble of EOF code	
10	First byte of first data pair	First byte is ASCII code for first hex digit; second byte is ASCII code for second hex digit.
11	Second byte of first data pair	
$8 + (N*2)$	First byte of last data pair	Last pair of data characters
$9 + (N*2)$	Second byte of last data pair	



Model II TRSDOS

CHARACTER NUMBER	CONTENTS	COMMENTS
10 + (N*2)	First byte of data checksum (high nibble)	This pair represents 2's complement of the data (all byte pairs after the ":" up to but not including the checksum). Note that each byte pair is converted back to the original byte of data before it is summed.
11 + (N*2)	Second byte of data checksum (low nibble)	

Sample record:

CHARACTER NUMBER	SAMPLE DATA	
	ASCII	HEX VALUE
1	“.”	3A
2	“0”	30
3	“2”	32
4	“2”	32
5	“8”	38
6	“0”	30
7	“0”	30
8	“0”	30
9	“0”	30
10	“3”	33
11	“7”	37
12	“7”	37
13	“0”	30
14	“5”	35
15	“D”	44

This record will contain 2 byte-pairs of data:

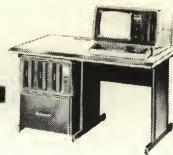
“3” “7” representing the value X'37'

“7” “0” representing the value X'70'

and will start loading at X'2800'. The one-byte sum of the original bytes (represented in pairs by characters 2 through 13) is X'A3'. The 2's complement of X'A3' is X'5D'—which is represented in bytes 14 and 15.

Notes on Using The Upload Function

Because of the baud rate and the absence of complex protocols for error handling, re-transmissions, etc., the Upload function is intended for hard-wired machines, e.g., from a development machine to the Model II in the immediate vicinity. For information on Model II to Model II connections, see RS232C in Section 4.



DIR

List the Diskette Directory

DIR:*d* SYS,PRT

d is a drive specification. (The colon : before *d* is optional.) If *d* is omitted, drive 0 is used.

SYS tells TRSDOS to list system and user files. If SYS is omitted, only user files are listed.

PRT tells TRSDOS to list the directory to the Printer. If PRT is omitted, TRSDOS lists the directory on the Console Display.

This command gives you information about a diskette and the files it contains. it contains.

To pause the listing, press **HOLD** . To continue, press **HOLD** again. To terminate the listing, press **ESC** .

Examples

DIR

Displays the directory of user files in drive 0.

DIR 1 PRT

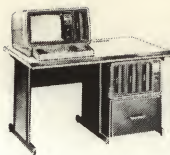
Lists to the Printer the directory of the user files in drive 1.

DIR { SYS,PRT }

Lists to the Printer the directory of System and user files.

The braces are required to prevent TRSDOS from taking SYS as an invalid drive specification.

An illustration of a sample directory is on the next page.

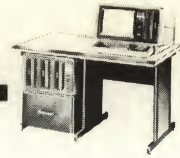


Sample Directory Listing

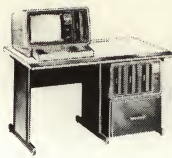
1	2	3	4	5	6	7	8	9	10	11
DISK NAME: EXAMPLES				DRIVE: 2		10/04/79		12-5-01		
FILE NAME	CREATED	ATTRB	FILE REC	TYPE LEN	NMBR	NMBR	SPACE	EOF		
	MM DD YY				RECS	EXTS	ALLOC USED	BYTE		
SWAP/M2X	10 4 79	D*X0	F 1		496	1	5 2	239		
PEEK	10 4 79	P*X0	F 256		1	1	5 1	0		
PEEK/BAS	10 4 79	D*X0	F 1		77	1	5 1	76		
VDREAD	10 4 79	P*X0	F 256		****	0	0 ***	***		
VDREAD1/BAS	10 4 79	D*X0	F 1		1000	1	5 4	231		
ERRPRINT	10 4 79	P*X0	F 256		1	1	5 1	0		
M2BAS1C/EX	10 4 79	D*X0	F 1		377	1	5 2	120		
*** 369 FREE GRANULES IN 2 EXTENTS ***										

What the column headings mean

- ① **Disk Name**—The name assigned to the diskette when it was formatted.
- ② **File Name**—The name and extension assigned to a file when it was created. The password (if any) is not shown.
- ③ **Creation Date**—When the file was created.
- ④ **Attributes**—A four-character field.
 The first character is either P for Program file or D for Data file.
 The second character is either S for System file or * for User file.
 The third character gives the password protection status.
 X The file is unprotected (no passwords).
 A The file has an access word but no update word.
 U The file has an update word but no access word.
 B The file has both update and access words.
 The fourth character specifies the level of access assigned to the access word:
 - 0,1 Kill file and everything listed below.
 - 2 Rename file and everything listed below.
 - 3 Not used
 - 4 Write and everything listed below.
 - 5 Read and everything listed below.
 - 6 Execute only.
 - 7 None.



- ⑤ **File Type**—Indicates the record type for the file.
 - F Fixed-length records.
 - V Variable-length records.
- ⑥ **Record Length**—Assigned when the file was created (applies to fixed-length record files only).
- ⑦ **Number of Records**—How many logical records have been written. Asterisks signify none have been written or file has variable length records and number written cannot be calculated. If number exceeds 65535, it starts over at zero. That is, it is a modulo 65536 number. True number of records can be inferred from Space Used column.
- ⑧ **Number of Extents**—How many segments (contiguous blocks of up to 32 granules) of disk space are allocated to the file. Asterisks signify none have been allocated.
- ⑨ **Space Allocated**—How many sectors (256 byte blocks) are allocated to the file. Asterisks signify none have been allocated.
- ⑩ **Space Used**—How many of these sectors have actually been written. Asterisks signify none have been allocated.
- ⑪ **End of File (EOF) Byte**—Shows the starting position in a sector of the last record written. Will be in last sector unless last record was spanned.
- ⑫ **Free Space Remaining**—Tells how many granules (1280-byte blocks) are free for storing new information. Also tells how the free space is organized, i.e., how many contiguous blocks (extents) make up the free space.



DO

Begin Auto Command Input from Disk File

DO file
file specifies a file created with the BUILD command

This command reads and executes the lines stored in a special-format file created with the BUILD command. The System executes the commands just as if they had been typed in from the Keyboard, except that they are not echoed to the Video Display (except for PAUSE).

Command lines in a BUILD file may include library commands or file specifications for user programs.

When DO reaches the end of the automatic command input file, it returns control to TRSDOS.

The DEBUG command cannot be included in an automatic command input file.

Running User Programs from a DO-file

In addition to executing TRSDOS library commands, you can load and execute user programs from a DO-file. You will probably want to make your program name be the last line in the DO-file (see **Note**). Before the DO-processor starts the last line in the DO-file, it shuts off certain special functions so that your program may execute normally (see **Note** below). For example, if you wanted to perform some library commands and then run a BASIC program called MENU, you would make this the last line in your program:

BASIC MENU

You can also “chain” do files, by putting another DO command at the end of a DO-file. The DO command *must* be the last line in the DO-file.

Note: You can run user programs from the middle of a DO-file. The program will run normally, with one important exception: pressing **BREAK** while your program is executing will interrupt your program, terminate DO-file processing, and return you to TRSDOS. Furthermore, your user program cannot set up a **BREAK** -processing program (see SETBRK).



Examples

DO STARTER

TRSDOS will begin automatic command input from STARTER, after the operator answers the Date and Time prompts.

AUTO DO STARTER

Whenever you start TRSDOS, it will begin automatic command input from STARTER.

Sample Use

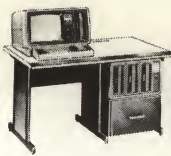
Suppose you want to set up the following TRSDOS functions automatically on start-up:

```
FORMS W=80  
CLOCK ON  
VERIFY OFF
```

Then use BUILD to create such a file. If you called it BEGIN, then use the command:

AUTO DO BEGIN

to perform the commands each time TRSDOS starts up.



DUMP

Store a Program Into a Disk File

```
DUMP file START=address-1, END=address-2, TRA=address-3  
      RELO=address-4, RORT=letter
```

file is a file specification.

START=*address-1* specifies the start address of the memory block.

END=*address-2* specifies the end address of the memory block.

TRA=*address-3* specifies the transfer address, where execution starts when the program is loaded. If omitted, *address-4* is used.

RELO=*address-4* specifies the start address for loading the program back into memory. If omitted, *address-1* is used.

RORT=*letter* specifies whether the program is directly executable from TRSDOS. RORT stands for "RETURN OR TRANSFER". If RORT=R, then TRSDOS can load but not execute the file. If RORT=T, then TRSDOS can load and execute the file from the TRSDOS READY mode. If RORT is omitted, RORT=T is used.

Note: Addresses must be hexadecimal form, without the x' notation.

This command copies a machine-language program from memory into a program file. You can then load and execute the program at any time by entering the file name in the TRSDOS READY mode.

You can enter machine language programs directly into memory, via the DEBUG command.

Examples

```
DUMP LISTER/CMD START=7000, END=7100, TRA=7004
```

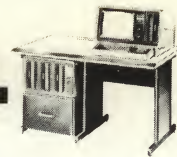
Creates a program file named LISTER/CMD containing the program in memory locations X'7000' to X'7100'. When loaded, LISTER/CMD will occupy the same addresses, and TRSDOS will protect memory beginning at X'7000'. The program is executable for the TRSDOS READY level.

```
DUMP PROG2/CMD START=6000, END=6F00, TRA=3010, RELO=3000
```

Creates a program file named PROG2/CMD containing the program in addresses X'6000' to X'6F00'. When loaded, PROG2/CMD will reside from X'3000' to X'3F00'. Execution will start at X'3010'. The program is executable from TRSDOS READY.

```
DUMP ROUTINE/1 START=6F00, END=6FFF, RORT=R
```

Creates a program file which cannot be executed from the TRSDOS READY level. Typically, this would be a routine to be called by another program.



ERROR

Display Error Message

ERROR number
number is a decimal number for a TRSDOS error code.

This command displays a descriptive error message. When TRSDOS gives you a reverse (black-on-white) message like:

```
* * ERROR 47 * *
```

You type back

```
ERROR 47
```

to see the full error message.

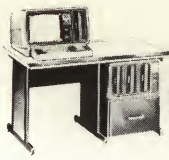
Example

```
ERROR 3
```

Gives you the message

```
PARAMETER ERROR ON CALL
```

For a complete list of error codes, messages and explanations, see the Appendix to this manual.



FORMS Set Printer Parameters

FORMS {P=*page size*, L=*lines*, W=*width*, C=*control*, S, R }

FORMS {T}

P = *page size* tells TRSDOS the total number of lines per page. If omitted, 66 is used.

L = *lines* tells TRSDOS the maximum number of lines to print before an automatic form feed. If omitted, 60 is used. *lines* cannot be greater than *page size*.

W = *width* tells TRSDOS the maximum number of characters per line. If omitted, 132 is used.

C = *control* tells TRSDOS to initialize the Printer by sending it the specified code. The code can be any hexadecimal value in the range [0,'FF']. Do not use the x" notation.

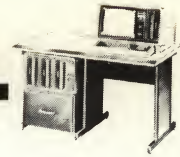
S = selects the serial printer driver. If you are going to use this option, you must first initialize channel B with SETCOM.

R = restores the parallel printer driver. If both S and R are omitted, R is assumed. In other words, the parallel printer driver is the default printer driver.

FORMS T is a special version of this command, telling TRSDOS to advance printer to top of form. Printer must have been previously initialized and must be ready. When T is given in the option list, any other keywords in the option list are ignored.

This command lets you set up the TRSDOS Printer software to suit the Printer you have attached. If the Printer was ready when you started TRSDOS, and the default parameters P=66, L=60, W=132, and C=0 are appropriate, then you do not need to use this command.

In addition to setting parameters, FORMS verifies that the Printer is ready, and it lets you adjust paper to the top of form.



Examples

`FORMS`

Resets all parameters to their default values.

`FORMS L=56`

Resets the maximum number of printed lines per page to 56, leaving 10 lines blank on each page.

`FORMS C=14`

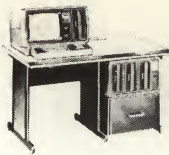
Sends the initialization code X'14' to the Printer.

`FORMS T`

Advances Printer to top of form. Useful when you have done some printing and want to start next printing at top of form.

`FORMS W=80, S`

Sets up the serial printer driver with 80-character lines and all other parameters according to their default values.



MODEL II TRSDOS

Setting the Parameters

Page Size. Multiply your form length in inches by the number of printed lines per inch to get the appropriate value. Most Printers print 6 lines per inch. Therefore standard 11-inch forms have a page size of 66 lines. That's why the default is `PAGE=66`.

Lines per page. This number determines the number of blank lines at the bottom of each page. If you set *lines* equal to *page size*, then TRSDOS will print every line on the page. If you set *lines* equal to *page size* minus 6, then TRSDOS will leave 6 blank lines on each page. Lines per page cannot exceed *page size*.

Width. This number sets the maximum number of characters per line. If a print line exceeds this width, TRSDOS will automatically break the line at the maximum length and continue it at the beginning of the next Print line.

Control Codes. Some Printers require an initialization code (for example, to set up for double-size characters). The code you specify is sent to the Printer during execution of the FORMS command.

Using a Serial Printer

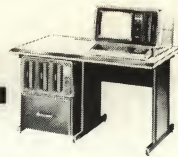
The serial printer driver uses channel B on the back panel of the display console. Connect your printer to this channel. Radio Shack's RS-232-2 Cable, Catalog Number 26-4403, will work with many serial printers. If channel A is not connected, place the serial terminator plug on that channel.

Before initializing the serial printer driver with FORMS, you must connect the printer and execute the SETCOM command with parameters appropriate for your printer. For example, if your printer uses 300 baud, 7-bit words, no parity and 1 stop bit, you would use a command like this:

```
SETCOM B=(300, 7, N, 1)
```

Then you would execute the FORMS command. For example, if your serial printer had a maximum line width of 80 characters, you might use this command:

```
FORMS W=80, S
```

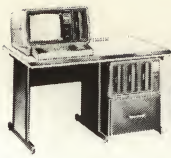



Technical Information

The serial printer driver uses the following pins of channel B (refer to the Model II Operation Manual for a pin diagram):

Signal Name	Pin #
GROUND	1, 7
DATA SET READY	6
CLEAR TO SEND	5
CARRIER DETECT	8
TRANSMIT DATA	2
REQUEST TO SEND	4
DATA TERMINAL READY	20

Note: If your serial printer does not support the CLEAR TO SEND signal, connect pins 5 and 20 on channel B. If it does not support the DATA SET READY signal, connect pins 6 and 20 on channel B.



FREE Display Disk Allocation Map

FREE :*d* PRT

:*d* is a drive specification. (The colon : before *d* is optional.) If :*d* is omitted, drive 0 is used.

PRT tells TRSDOS to send the map to the Printer. If PRT is omitted, TRSDOS sends the map to the Console Display.

This command gives you a map of granule allocation on a diskette. (A granule, 1280 bytes, is the unit of space allocation.) This information is useful when you want to optimize file access time.

When a diskette has been used extensively (file updates, files killed, extended, etc.), files often become segmented (dispersed or fragmented). This slows the access time, since the disk read/write mechanism must move back and forth across the diskette to read or write to a file.

FREE helps you determine just how segmented your disk files are. If you decide that you'd like to re-organize a particular file to allow faster access, you can then COPY it onto a relatively "clean" diskette.

Example

FREE

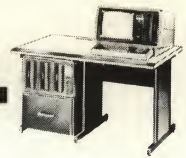
Displays a free space map of the diskette in drive 0.

FREE {PRT}

Lists the free space map for drive 0 to the Printer. The braces are required in this example, since no drive specification is included. Otherwise TRSDOS would take PRT as an invalid drive specification.

FREE 2 PRT

Lists the Drive 2 map to the Printer.

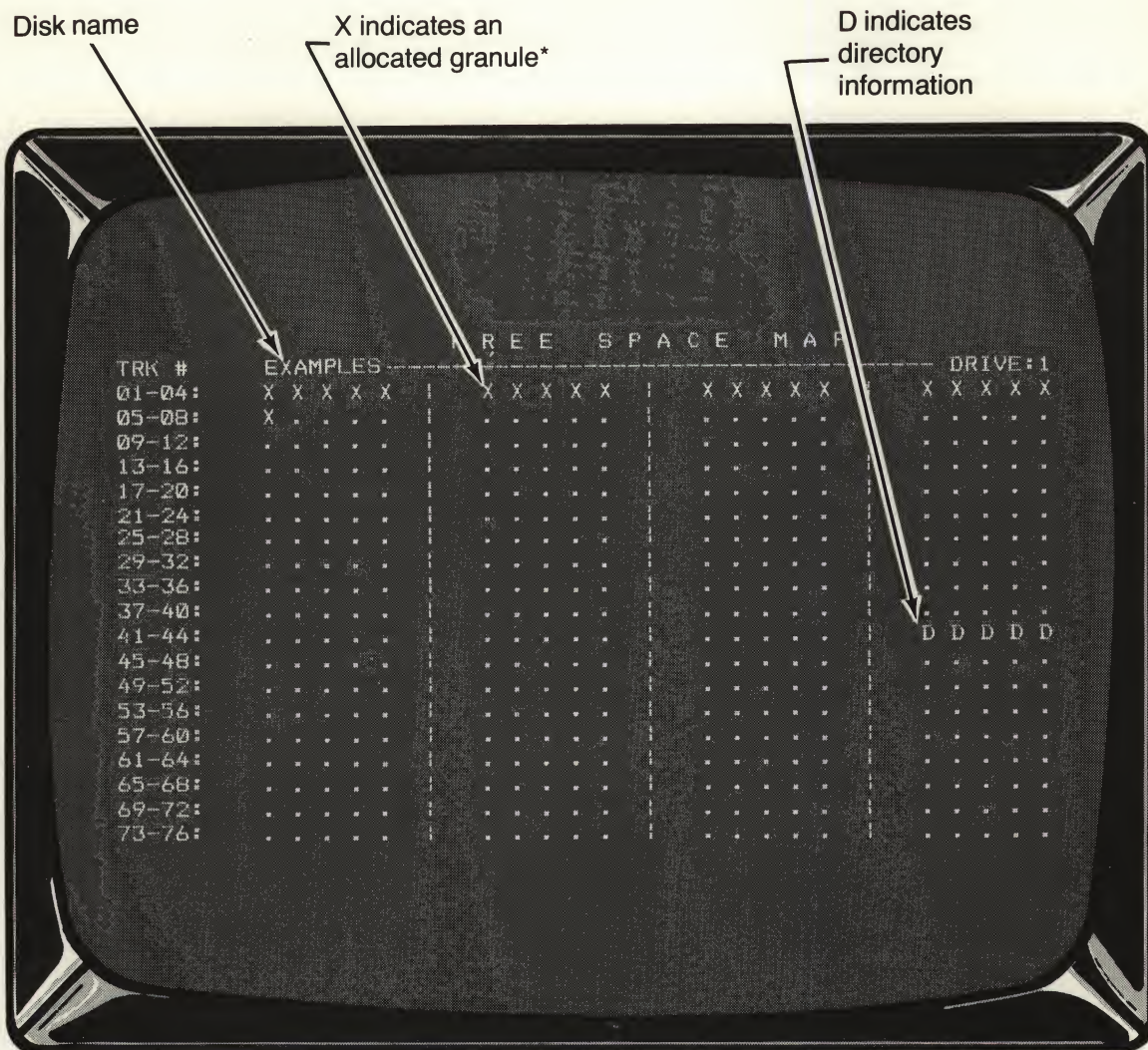


A Typical FREE Display

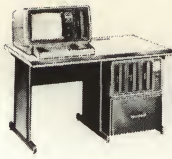
Four special symbols are used in the FREE map:

- . Unused Granule
- D Directory Information
- X Allocated Granule
- F Granule Contains a Flawed Sector (Unusable)

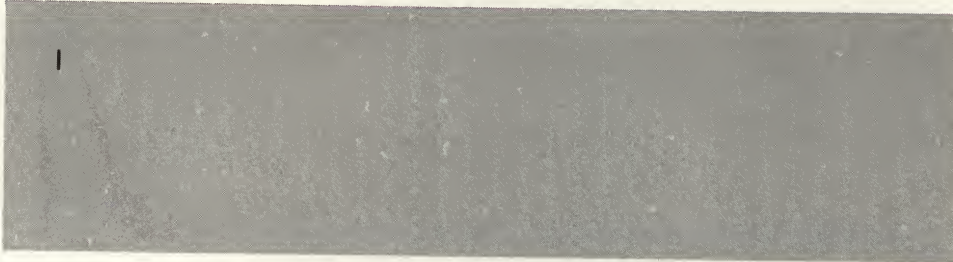
Here's a typical display:



*In this map, tracks 1-4 are fully allocated; 1 granule in track 5 is allocated; and the directory takes all of track 40.



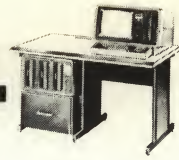
I **Swap Diskettes**



Immediately after you change diskettes in any drive, enter this command so TRSDOS will be able to perform important “bookkeeping” tasks.

Example

I
tells the System you have changed one of the diskettes.



KILL

Delete a File

KILL file
file is a file specification.

This command deletes a file from the directory and frees the space allocated to that file. If no drive is specified, TRSDOS will search for the file, starting with drive 0. Before deleting the file, TRSDOS will display the file name and the drive that contains the file. Type Y **ENTER** to Kill the file, N **ENTER** to cancel the command.

Examples

KILL TESTPROG/BAS

Deletes the named file from the first drive that contains it.

KILL JOBFIL/IDY.f099Y

Deletes the named file from the first drive that contains it. The file is protected with the password foggy.

KILL FORM/123:3

Deletes FORM/123 from drive 3.

Sample Uses

When updating a file, it is a good practice to input from the old file and output updated information to a new file. That way, if the update is wrong, you still have the old file as a backup. When you have verified that the update file is correct, you can Kill the old file.

KILL is also useful in conjunction with pre-allocated files. Suppose you have finished writing to a pre-allocated file, and one or more granules are unused in the pre-allocated file. Then you can copy the pre-allocated file to a dynamically allocated file, and afterwards Kill the pre-allocated file. This is the only way to reduce the size of a pre-allocated file.



LIB

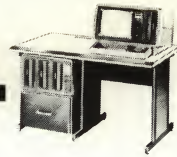
Display Library Commands

LIB

This command lists to the Display all the Library Commands.

Example

LIB



LIST

List Contents of a File

LIST *file* { PRT, SLOW, R=*record-number* }

file is a file specification.

PRT tells TRSDOS to list to the Printer. If PRT is omitted, the Console Display is used.

SLOW tells TRSDOS to pause briefly after each record. If omitted, the listing is continuous.

R=*record number* tells TRSDOS the starting record for the listing. *record number* must be in the range [1,65535]. If omitted, record 1 is used.

This routine lists the contents of a file. The listing shows both the hexadecimal contents and the ASCII characters corresponding to each value. For values outside the range [X'20',X'7F'], a period is displayed.

To stop the listing, press **HOLD**. Press **HOLD** again to continue. Press **ESC** or **BREAK** to terminate the listing.

Examples

LIST DATA/BAS

Lists the contents of DATA/BAS.

LIST TEXTFILE/1 SLOW

Lists the contents of TEXTFILE/1, pausing after each record.

LIST TEXTFILE/1 R=100, A

The listing starts with the 100th record in TEXTFILE/1. Only ASCII characters are displayed.

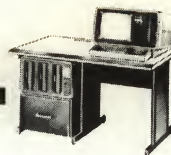
LIST PROGRAM/CMD PRT

Lists the file PROGRAM/CMD to the Printer.



LIST numbers each record as it is listed, and prints a heading showing the relative position of each byte in the record. Here's a sample listing after the command:

[illegible]



LOAD

Load a Program File

LOAD *file*

file is a file specification for a file created by the DUMP command.

This command loads into memory a machine-language program file. After the file is loaded, TRSDOS returns to the TRSDOS READY mode.

You cannot use this command to load a BASIC program or any file created by BASIC. See the BASIC Reference Manual for instructions on loading BASIC programs.

Example

```
LOAD PAYROLL/Pt1
```

Sample Use

Often several program modules must be loaded into memory for use by a master program. For example, suppose PAYROLL/pt1 and PAYROLL/pt2 are modules, and MENU is the master program. Then you could use the commands:

```
LOAD PAYROLL/Pt1
LOAD PAYROLL/Pt2
```

to get modules into memory, and then type:

```
MENU
```

to load and execute MENU.

If PAYROLL/pt1 and PAYROLL/pt2 were Dumped with RORT=R, then you can load by typing the file name without the LOAD command, i.e.,

```
PAYROLL/Pt1
PAYROLL/Pt2
```

After each is loaded, TRSDOS READY returns.



PAUSE

Pause Execution for Operator Action

PAUSE prompting message

prompting message is an optional message to be displayed during the pause.

This command is intended for use inside a DO file. It makes TRSDOS print a message and then wait for the operator to press **ENTER**.

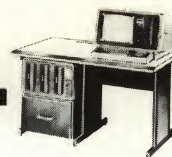
Example

```
PAUSE                Insert Diskette #21
```

Prints PAUSE followed by the message and prompts the operator to press **ENTER** to continue.

```
PAUSE
```

Prints PAUSE and prompts the operator to press **ENTER** to continue. See BUILD and DO for sample uses.



PURGE

Delete Files

PURGE :d {*file-class*}

:d is drive specification. The colon : is optional. If :d is omitted, drive 0 is used.

file-class is one and only one of the following:

SYS	System files (program and data)
PROG	User machine-language program files
DATA	User data files
ALL	All files, user and system

If *file-class* is omitted, DATA is used.

This command allows quick deletion of files from a particular diskette. To use PURGE, you must know the diskette's master password. (TRSDOS System diskettes are supplied with the password PASSWORD.)

All System files are required for TRSDOS to function. Do not eliminate System files if you want to use the diskette in drive 0.

When the command is entered, TRSDOS will ask for the diskette's password. Type in up to 8 characters, and press **ENTER** if you typed fewer than 8 characters. The System will then display user file names one at a time, prompting you to Kill or leave each file.

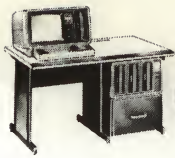
Example

PURGE 1

TRSDOS will let you purge data files from drive 1. **This would include BASIC programs.**

PURGE

TRSDOS will let you purge data files from drive 0.



PROT Use Diskette's Master Password

PROT *:d* { *OLD=password, options* }

:d is a drive specification. The colon is optional.
OLD=password specifies the diskette's current master password.
(Your TRSDOS diskette is supplied with the password **PASSWORD**.)
options include any of the following:
PW, NEW=password Gives TRSDOS the new master password (up to eight alphanumeric characters). **PW** and **NEW=password** must be used together.
LOCK Tells TRSDOS to protect all user files with the latest master password. Update and access words will both be set to this password.
UNLOCK Tells TRSDOS to remove passwords from all user files.

If **LOCK** and **UNLOCK** are omitted, user file protection is left unchanged. If one is used, the other must be omitted.

PROT changes file protection on a large scale. If you know the diskette's master password, you can change it. You can also protect or un-protect all user files.

A diskette's master password is initially assigned during the format or backup process. The TRSDOS diskette is supplied with the master password **PASSWORD**.

Example

```
PROT 1 OLD=PASSWORD, PW, NEW=H2O
```

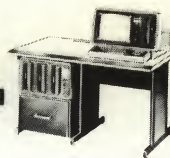
Tells TRSDOS to change the master password of the drive 1 diskette from **PASSWORD** to **H2O**.

```
PROT 0 OLD=H2O, UNLOCK
```

Tells TRSDOS to remove passwords from every user file on the drive 0 diskette (must have the password **H2O**).

```
PROT 0 OLD=H2O, PW, NEW=ELEPHANT, LOCK
```

Tells TRSDOS to change the master password from **H2O** to **ELEPHANT** and assign the new one to every user file.



RENAME

Rename a File

RENAME *file-1* TO *file-2*

file-1 and *file-2* are file specifications. If *file-2* includes a drive specification or password, it will be ignored. The file will retain its former password, if any.

file-1 and *file-2* are separated by a slash (/). A comma or space may also be used.

This command lets you rename a file. Only the name/extension is changed; the data in the file and its physical location on the diskette are unaffected.

RENAME cannot be used to change a file's password. Use ATTRIB to do that.

Examples

RENAME Miss/BAS TO Ms/BAS

TRSDOS will search for Miss/BAS starting with drive 0, and will rename it to Ms/BAS.

RENAME REPORT/AUG:3 TO REPORT/SEP

Renames REPORT/AUG on drive 3 to REPORT/SEP.

RENAME MASTER.12345678 TO MASTER/A

Searches for MASTER and renames it to MASTER/A. The password 12345678 must grant at least RENAME access (see Passwords in chapter 1). The renamed file has the same password.



SETCOM

Set Up RS-232C Communications

SETCOM { A=(*baud rate, word length, parity, stop bits*),
B=(*baud rate, word length, parity, stop bits*) }
A=(*options*) tells TRSDOS to initialize channel A.
To turn channel A off, use A=OFF instead of A = (*options*)
If A = (*options*) is omitted, status of channel A is unchanged.
B=(*options*) tells TRSDOS to initialize channel B.
To turn channel B off, use B=OFF.
If B = (*options*) is omitted, status of channel B is unchanged.
The options tell TRSDOS what RS-232C parameters to use. The following parameters are available:

<i>baud rate</i>	110, 150, 300, 600, 1200, 2400, 4800, 9600 If not specified, 300 is used.
<i>word length</i>	5, 6, 7, 8 If not specified, 7 is used.
<i>parity</i>	E for even, O for odd, N for none If not specified, even is used.
<i>stop bits</i>	1, 2 If not specified, 1 is used.

Every option but the last must be followed by a comma. The options are positional, e.g., the third item in an option list must always specify parity. To use a default value, omit the option. If you want to list subsequent options, you must include a comma for each default.

This command initializes RS-232C communications via channels A and B on the back panel. Before executing it, you should connect the communications device (modem, etc.) to the Model II.

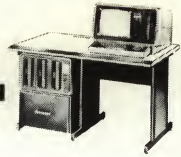
To **change** the settings on a currently active channel, you must first turn the channel **off**. If the channel is already off when you try to turn it off, you'll get an error message.

See the Model II Operation Manual for a description of RS-232C signals used in channels A and B. For hard-wired connection from one Model II to another, see the wiring diagram in Technical Information, RS232C supervisor call.

SETCOM uses the Special Programming Area above TOP (see Memory Requirements). To use the serial I/O channels from BASIC you must execute SETCOM **before** starting BASIC.

Once you initialize a channel, you can begin sending and receiving data, using four system routines that are set up during initialization:

ARCV	Channel A receive, function code 96
ATX	Channel A transmit, function code 97
BRCV	Channel B receive, function code 98
BTX	Channel B transmit, function code 99



These system routines are only available when the respective channel has been initialized, See **Technical Information** for details.

Examples

```
SETCOM A=( )
```

Sets up channel A for serial communications, using all the default parameters. System function calls 96 and 97 are available for serial I/O. The status of channel B is unchanged.

```
SETCOM B=(4800, B, , 2), A=OFF
```

Sets up channel B:

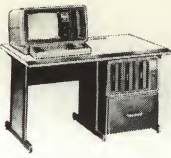
baud rate	4800
word length	8 bits
parity	Even (default)
stop bits	2

and turns off channel A.

```
SETCOM A=(2400, B, 0), B=( , , , 2)
```

Sets up channels A and B:

	Channel A	Channel B
baud rate	2400	300 (default)
word length	8	7 (default)
parity	Odd	Even (default)
stop bits	1 (default)	2



TIME Reset or Get the Time

TIME *hh.mm.ss*

hh is a two-digit hour specification.

mm is a two-digit minute specification.

ss is a two-digit second specification. If *.ss* is omitted, *.00* is used.

If *hh.mm.ss* is given, TRSDOS resets the time.

If *hh.mm.ss* is not given, TRSDOS displays the current time and date.

This command lets you reset the time or display the date and time.

The operator sets the time initially when TRSDOS is started up. After that, TRSDOS updates the time and date automatically, using its built-in clock and calendar.

When you request the time, TRSDOS displays it in this format:

```
THU JUL 19 1979 200 --- 14.15.31
```

for Thursday, July 19, 1979, the 200th day of the year, 2:15:31 pm.

Note: If the time passes 23:59:59, TRSDOS does not start over at 00:00:00. Instead, it continues with 24:00:00. However, the next time you use the TIME or DATE command, the time will be converted to its correct 24-hour value, and the date will be updated. If the clock is allowed to run past 59.59.59, it will re-cycle to zero, and the date will not be updated to include the 60-hour period.

Examples

```
TIME
```

Displays the current date and time.

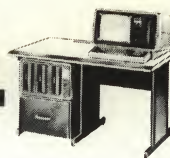
```
TIME 13.20.00
```

Resets the time to 1:20:00 pm.

```
TIME 18.24
```

Resets the time to 6:24:00

Note: Periods are used instead of the customary colons since periods are easier to type in — you don't have to press **SHIFT**



VERIFY

Automatic Read After Write

VERIFY {*switch*}

switch is one of the following:

ON Turn on the verify function.

OFF Turn off the verify function.

If *switch* is omitted, the current status is displayed and is left unchanged.

This command controls the verify function. When it is on, TRSDOS will read after each write operation, to verify that the data is readable. If the data is not readable, after retries, TRSDOS will return an error message, so you'll know that the operation was not successful.

Note: TRSDOS always verifies directory writes. User writes (writing data into a file) are only verified when VERIFY is ON.

TRSDOS starts up with VERIFY ON. For most applications, you should leave it ON.

Examples

VERIFY ON

Turns on the verify function.

VERIFY OFF

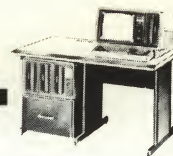
Turns off the verify function.

VERIFY

Displays the status of the verify switch.

Section 3

Utility Programs



Introduction to Utility Programs

NAME	PURPOSE
BACKUP	Duplicate a Diskette
FORMAT	Organize a Diskette
PATCH	Change contents of a disk file
TERMINAL	Communications program
XFERSYS	Transfer operating system

Several other "incidental" utilities are included: BASCOM, COMSUB and DOCOM (all for serial communications); EXDATM and DATM (for date calculations); HERZ50 (for non-USA users).



BACKUP

Duplicate a Diskette

BACKUP

This program duplicates the data from a diskette onto another **previously formatted** diskette, by copying all allocated granules.

To execute BACKUP, type:

BACKUP

BACKUP provides all necessary prompting for input. The destination diskette must be formatted; if it contains data, BACKUP will warn you and ask if you want to write over the data.

Prompting Messages

SOURCE DRIVE?

Type in the number of the drive which will contain the source diskette (diskette to be duplicated). This can be any drive 0 through 3. Type Q to quit.

SOURCE DISK PASSWORD?

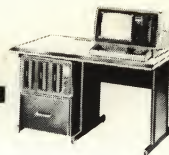
Type in the password, up to 8 numbers or letters, and press ENTER if you typed fewer than 8. **The original password on your TRSDOS diskette is:** PASSWORD.

DESTINATION DRIVE

Type in the number of the drive which will contain the destination diskette. This can be any drive 0 through 3.

DO YOU WANT TO CHANGE DISK INFORMATION?

Type Y if you want to change the master password, diskette name, or date. BACKUP will prompt you to enter the new information (if you don't want to change any of these, type N).



- To change the password, type in up to 8 numbers and letters, and press **ENTER** if you typed fewer than 8.
- To change the diskette name, type in up to 8 numbers and letters and press **ENTER** if you typed fewer than 8.
- To change the date, obey the prompts.

DESTINATION DISK READY?

Insert the destination diskette into the destination drive and type Y.

Note: For single-drive backups, you will need to swap source and destination diskettes when prompted by the messages:

SOURCE DISK READY? and DESTINATION DISK READY?

If there is a flaw on one of the **destination** tracks, BACKUP will terminate and return to TRSDOS without completing the duplication. The flawed diskette should be re-formatted and used as a data diskette (multi-drive users only).

BACKUP also does a consistency check of the directory track, comparing individual file entries with a separate table of space allocation. If any inconsistencies are found, BACKUP lists the affected files and prompts you to choose one of the following options:

- 1 Copy only those files whose allocation information is consistent.
- 2 Copy all files, regardless of inconsistencies.
- 3 Abort the backup procedure.

Files with inconsistencies may or may not be usable.



FORMAT

Organize a Diskette

FORMAT:*d* {*ID=diskette name*, *PW=password*, *option*}

d specifies the drive to be used. The colon : is optional. *d* can specify any drive 0 through 3.

ID=diskette name tells TRSDOS what name to assign.
diskette name can be up to 8 numbers and letters with no embedded blank spaces.

PW=password tells TRSDOS what password to assign.
password can be up to 8 numbers or letters with no embedded blank spaces.

option tells TRSDOS how much verifying to do:

FULL	Thorough check for flaws.
QUICK	Quick check for flaws.
NONE	No checking for flaws.

If *option* is omitted, FULL is used.

This program organizes a diskette into tracks and sectors. The diskette may be either blank (new or bulk-erased) or previously formatted. If it contains data, FORMAT will warn you and ask if you want to continue. If you continue, the data will be lost. In general, it's a good practice to bulk erase a diskette before formatting it.

Format also does a specified amount of checking for areas on the disk which cannot store data due to flaws in the recording surface. If it finds a flawed area, TRSDOS "locks out" the affected track and will never try to write to that track.

Examples

```
FORMAT :1 { ID=ACCOUNTS, PW=mouse }
```

Drive 1 will be used, and the disk will be given the name ACCOUNTS and master password mouse. TRSDOS will do the full test for flaws.

```
FORMAT :0 { ID=TESTISK, PW=PASSWORD, QUICK }
```

Drive 0 will be used, and the diskette will be given the name TESTDISK and master password PASSWORD. FORMAT will do a quick check for flaws.

```
FORMAT :3 { ID=Jack, PW=1234578, NONE }
```

Drive 3 will be used, diskette name Jack, password 12345678, with no checking for flaws.



When to Format

To prepare a new diskette.

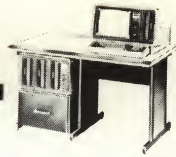
Before you can use a new diskette, you must format it. After formatting, record the disk name, date of creation and password in a safe place. This will help you estimate how long a diskette has been in use, and prevent your forgetting the master password. (For this application, always use the FULL verify option.

To erase all data from a diskette.

To “start over” with a diskette, you can format it. All data will be lost. For this application, the QUICK verify option is probably adequate—unless you have had problems disk input/output errors with the diskette.

To lock out flawed areas.

After long use, flaws may develop on a diskette. Reformat the diskette to lock out these tracks while leaving the good tracks available for data storage. Use the FULL verify option for this application.



BASCOM (BASIC Program) COMSUB (USR Subroutine) DOCOM (TRSDOS DO-File) Serial I/O Demonstration

BASCOMnn

is the name of the BASIC program; *nn* = 32 for 32K computers; *nn* = 64 for 64K computers.

COMSUBnn

is the name of the USR subroutine; *nn* = 32 or 64 as above.

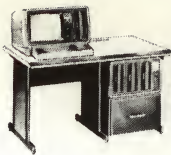
DOCOMnn

is the name of a TRSDOS DO-file that automatically initializes channel A for serial I/O, loads COMSUBnn and starts BASIC; *nn* = 32 or 64 as above.

The BASIC communication program BASCOM and the communications subroutine COMSUB together perform a "terminal" function, demonstrating an application for Model II's serial interface. The programs are provided to give you a feel for interfacing programs at the assembly-language level with BASIC programs. BASCOM calls the machine-language COMSUB via the USR function. Like TERMINAL, BASCOM and COMSUB are included on your system diskette and may be examined by employing the TRSDOS command LIST (or the BASIC command LIST in the case of BASCOM).

The programs allow you to use the keyboard of the Model II to send data in the form of ASCII characters to another computer or device; at the same time, characters transmitted on the other device will be received by the Model II and printed on the Display.

Serial channel A is used for sending and receiving. (You must put a terminator plug on channel B.) The programs alternately check channel A for a character received, and the keyboard for a character typed. Characters typed will be automatically echoed to the Video Display, though this can be defeated by making a two-byte modification to COMSUB; the NOPs at X'6F9F' and X'6FA0' (COMSUB32) or X'EF9F' and X'EFA0' (COMSUB64) should be modified to a LD (HL), 00.



MODEL II TRSDOS

Sample Use

Before using the two programs, make sure that channel A is connected to a modem and that channel B is fitted with the serial terminator or connected to some other serial device (e.g., a serial printer). Then, under TRSDOS READY, type:

DO DOCOM32

or

DO DOCOM64

depending on whether you have a 32K or 64K computer. DOCOM32 and DOCOM64 name DO files which

1. Execute the SETCOM command (parameters are set to default values)
2. Load the appropriate COMSUB subroutine
3. Load BASIC and reserve enough memory for COMSUB. When the BASIC prompt appears on the screen, type:

RUN "BASCOM32"

or

RUN "BASCOM64"

The program will load and begin.

Error Handling

When a transmit or receive error is detected, COMSUB will print the word ERROR followed by an 8-bit error code. The leftmost digit represents bit 7; the right-most digit, bit zero. (The program will continue to attempt serial I/O.)

If bit 3 is on, then the modem carrier was lost. Check the telephone connection and all other connections.

If bits 0, 1, and 2 are all off, the error will be a receive error. See ARVC, page 4/78 in the TRSDOS manual, for further details on the error code.

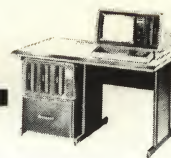
If bits 4, 5, 6, and 7 are all off, the error is a transmit error. See ATX, page 4/79 in the TRSDOS manual, for details.

Source Listing of COMSUB

The following fully commented listing is provided to aid assembly-language programmers in writing their own serial I/O routines.

Notice that the program is ORGed at X'EF80'. This is an appropriate address for 64K machines. For 32K machines, use a start address of X'6F80'.

UTILITY PROGRAMS



SUBROUTINE FOR BASIC COMMUNICATIONS PROGRAM

THIS ROUTINE MUST BE EXECUTED AT 300 BAUD OR HIGHER

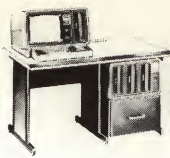
```

ORG      0EF80H      ;ON ENTRY DE POINTS TO A 3 BYTE STRING DESCRIPTOR
INC      DE           ;DE NOW POINTS TO LSB OF STRING ADDRESS
LD       A,(DE)       ;LSB OF STRING ADDRESS TO ACCUMULATOR
LD       L,A          ;LSB OF STRING ADDRESS TO REGISTER L
INC      DE           ;DE NOW POINTS TO MSB OF STRING ADDRESS
LD       A,(DE)       ;MSB OF STRING ADDRESS TO ACCUMULATOR
LD       H,A          ;MSB OF STRING ADDRESS TO REGISTER H
LD       A,(HL)       ;1 BYTE STRING TO ACCUMULATOR
CP       0             ;SEE IF CHARACTER IS ZERO
JR       NZ,XMITER    ;IF NOT ZERO TRANSMIT CHARACTER, ELSE FALL THROUGH TO RECIEVE CHARACTER
LD       A,96         ;SVC CALL:PORT A RECIEVE
RST      8            ;
JR       C,ERROR      ;QUIT ON ERROR IF MODEM CARRIER NOT PRESENT
RET      NZ           ;RETURN IF NO CHARACTER RECIEVED
OR       A            ;SET STATUS BITS
JR       NZ,ERROR     ;QUIT ON ERROR IF ANY STATUS BITS ARE SET
LD       (HL),B       ;PASS RECIEVED CHARACTER TO STRING LOCATION
RET

XMITER   LD       B,A      ;CHARACTER TO BE TRANSMITTED TO REGISTER B
LD       DE,0FFFFH      ;LOOP COUNT IF TRANSMITTER BUSY STATUS ENCOUNTERED
XMITI    LD       A,97     ;SVC CALL:PORT A TRANSMIT
RST      8            ;
JR       C,ERROR      ;QUIT UN ERROR IF MODEM CARRIER NOT PRESENT
NOP      *LD      (HL),00H ;HERE WHEN USING MODEM IN HALF-DUPLEX MODE
NOP      ;
RET      Z            ;RETURN IF CHARACTER TRANSMITTED
BIT      0,A          ;CHECK CLEAR TO SEND STATUS BIT
JR       NZ,ERROR     ;QUIT ON ERROR IF STATUS BIT SET
LD       A,D          ;MSB OF LOOP COUNT TO ACCUMULATOR
OR       E            ;LSB OF LOOP COUNT
DEC      DE           ;REDUCE LOOP COUNT
JR       NZ,XMITI     ;LOOP IF COUNT IS NOT ZERO, ELSE FALL THROUGH TO AN ERROR
ERROR    LD       (HL),00 ;DO NOT DISPLAY CHARACTER IF ERROR ENCOUNTERED
LD       B,8          ;LOOP COUNT (8 BIT STATUS BYTE)
LD       HL,BITST3    ;STORAGE AREA
BITEST   BIT       7,A   ;CHECK BIT 7 OF ACCUMULATOR FOR COMMUNICATIONS STATUS
LD       (HL),'0'      ;LOAD ASC-II ZERO
JR       Z,BITST1     ;JUMP IF STATUS BIT NOT SET
INC      (HL)          ;ASC-II ZERO => ASC-II ONE IF STATUS BIT SET
BITST1   RLCA        ;ROTATE ACCUMULATOR LEFT
INC      HL           ;MOVE TO NEXT STORAGE POSITION
DJNZ    BITEST        ;LOOP TO CHECK STATUS OF 8 BITS
LD       HL,BITST2    ;ERROR MESSAGE TO BE DISPLAYED
LD       B,14         ;LENGTH OF MESSAGE
LD       C,'.'        ;CHARACTER TO BE INSERTED AT THE END OF ERROR MESSAGE
LD       A,9          ;SVC CALL:VIDED LINE
RST      8            ;
RET

BITST2   DEFM      'ERROR' ;ERROR MESSAGE
BITST3   DEFS      8       ;STORAGE AREA FOR ERROR STATUS BITS TO BE DISPLAYED

```



DATM (Machine-Language Subroutine) EXDATM (Console Program) Date Calculations

DATMnn

is a machine-language subroutine that performs date calculations; *nn* = 32 for 32K computers; *nn* = 64 for 64K computers.

EXDATMnn

is a console program (can be executed from TRSDOS READY mode) that loads and uses DATMnn; *nn* = 32 or 64 as above.

EXDATM is a console routine which calls the date calculation program DATM and allows you to pass data to it from the keyboard.

DATM performs two functions:

1. Given a date and a timespan measured in days, the program adds the timespan to the date and calculates the resulting date. The timespan can't be greater than 65535 days; the year of the date must be at least 1600 and not larger than 9999.
2. Given two dates, the program calculates the number of days between the dates. The same parameter limitations apply.

Sample Uses

Under TRSDOS READY, type:

EXDATM64

or

EXDATM32

depending on whether you have a 64K or 32K machine. The program will ask you:

DATE ADDITION OR DIFFERENCE (A/D) . .

Date Addition

To add a date and a timespan, type:

A **ENTER**

The program will ask:

ENTER PARAMETERS

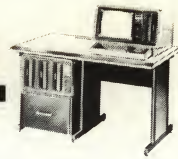
to which a sample reply might be:

05/14/1977 00180 **ENTER**

The program will compute and print the date which is 180 days after May 14, 1977:

THU NOV 10 1977 314

followed by the system time. The number 314 means that November 10 is the 314th day of 1977 (see the TRSDOS DATE command).



In general to find a new date from a date and a timespan, the date and timespan should be entered like this:

mm/dd/yyyy *jjjjj*

where *mm* is a two-digit number 01-12; *dd*, a two-digit number 01-31; *yyyy*, a four-digit number 1600-9999; and *jjjjj* (Julian measure of time), a five-digit number 00001-65535. (Julian time refers to the measure of time in days rather than hours, years, months, etc.) Be sure to use leading zeroes as required to make up the specified number of digits. And don't omit the space () between *yyyy* and *jjjjj*.

Date Difference

To find the number of days between two days, type D **ENTER** in response to the initial prompt (A/D). In response to the ENTER PARAMETERS prompt, type in your data as in this sample:

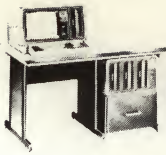
04/15/1979 12/31/1979 **ENTER**

The program will print the number of days between the dates you entered: 625.

In general to find a timespan between two dates, the dates should be entered like this:

mm/dd/yyyy *mm/dd/yyyy*

where *mm* is a two-digit number 01-12, *dd*, a two-digit number 01-31; and *yyyy*, a four-digit number 1600-9999. The earlier date must come first. Since the maximum timespan that can be returned is 65535 days, the two dates must not be separated by more than 179 years. Use leading zeroes where necessary to make up the specified number of digits, and don't omit the space () between the two dates.



DATM Source Listing For Assembly-Language Programmers

Notice that the program is ORGed at X'0000'. This is simply to make it easier for you to relocate the program. DATM32 is actually ORGed at X'6B70'; DATM64, at X'EB70'.

In general, the entry and exit conditions for this subroutine are the same as those given for EXDATM.

Entry Conditions

B = Function Switch

If B = 0 then compute date addition

If B \neq 0 then compute date difference

For date addition (B = 0)

(HL) = 16-byte text buffer located above X'27FF', as follows:

mm/dd/yyyyjjjjj

just like the input text for EXDATM described previously.

(DE) = 26-byte output buffer located above X'27FF', as follows:

NAME OF DAY	MONTH	DAY OF MONTH	YR.	DAY OF YR.	TIME	MONTH OF YR.	DAY OF WEEK
3	3	2	4	3	8	2	1

For date difference (B \neq 0)

(HL) = 21-byte text buffer located above X'27FF', as follows:

mm/dd/yyyyjjjjjmm/dd/yyyy

just like the input text for EXDATM described previously.

(DE) = 5-byte output buffer, located above X'27FF', as follows:

jjjjj

with leading blanks supplied as necessary.

To call DATM, execute a CALL to the start address.

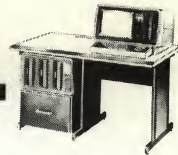
Exit conditions

(DE) = output text, in the format described previously.

NZ = Error occurred

A = Error code

UTILITY PROGRAMS

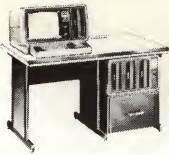


LOC OBJ CODE M STMT SOURCE STATEMENT PAGE 1
ASM 5.8

```

1  ;=====
2  ;
3  ; JULIAN DATE MATH ROUTINE
4  ;=====
5  ;
6  ; B=FUNCTION SWITCH
7  ;
8  ; IF B=Z THEN MM/DD/YYYY+JJJJJ=NEW DATE
9  ;
10 ; IF B=NZ THEN MM/DD/YYYY-MM/DD/YYYY=JJJJJ
11 ;
12 ; B=Z ENTRY : DE=>26 BYTE OUTPUT FIELD (SEE SYSTEM DATE ROUTINE)
13 ; HL=>MM/DD/YYYY JJJJJ
14 ;
15 ; MM=MONTH
16 ; DD=DAY
17 ; YYYY=YEAR (1600 <= YYYY <= 9999)
18 ; JJJJJ=JULIAN TIME SPAN IN DAYS (JJJJJ <= 65535)
19 ;
20 ; B=NZ ENTRY : DE=>5 BYTE OUTPUT FIELD (JJJJJ)
21 ; HL=>MM/DD/YYYY MM/DD/YYYY (EARLIER DATE, LATER DATE)
22 ;
23 ; EXIT : A=Z GOOD INDICATION
24 ; A=NZ A CONTAINS ERROR CODE
25 ;
26 ;=====
27 ;
28 ; FOR 64K VERSION RELOCATE AT EC60H
29 ;
30 ; FOR 32K VERSION RELOCATE AT 6C60H
31 ;=====
32 ;
33 ;
34 DATM PUSH HL ;SAVE CALLER'S REGISTERS
35 PUSH DE ;
36 PUSH BC ;
37 ;
38 LD A,H ;CHECK FOR BAD BUFFER ADDRESS
39 CP 28H ;
40 JR C,ERROR ;QUIT ON ERROR
41 LD A,D ;CHECK FOR BAD BUFFER ADDRESS
42 CP 28H ;
43 JR C,ERROR ;QUIT ON ERROR
44 LD (OUTDSS),DE ;SAVE OUTPUT BUFFER POINTER
45 XDR A ;ZERO ACCU.
46 CP B ;LOOK AT OPTION SWITCH
47 JP NZ,TWODAT ;ACT ON OPTION
48 JULDAT CALL EDIT ;EDIT INPUT
49 JR NZ,ERROR ;QUIT ON ERROR
50 LD A,' ' ;BLANK
51 CP (HL) ;CHECK BLANK
52 JR NZ,ERROR ;QUIT ON ERROR
53 INC HL ;BUMP INFORMATION POINTER
54 LD A,15H ;CONVERT
55 RST B ;SVC
56 LD (JULI),DE ;SAVE JULIAN SPAN
57 JR Z,JULT ;QUIT ON ERROR
58 ERROR LD A,3 ;PARAMETER

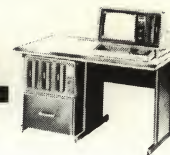
```



MODEL II TRSDOS

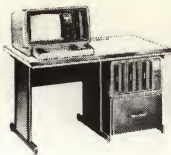
LOC	OBJ	CODE	M	STMT	SOURCE	STATEMENT	DATE	12/10/79--15:00	PAGE	2
							ASM	5.8		
002C	B7			59		OR	A		SET FLAGS	
002D	C1			60	ALEXIT	POP	BC		RESTORE REGISTERS	
002E	D1			61		POP	DE			
002F	E1			62		POP	HL			
0030	C9			63		RET			RETURN TO CALLER	
				64						
0031	CDF601	R		65	JULT	CALL	FIXFEB		LEAP YEAR ?	
0034	3A2202	R		66		LD	A,(MON1)		GET MONTH	
0037	1600			67	AGAIN	LD	D,0		ZERO MSB	
0038	5F			68		LD	E,A		MOVE MONTH	
003A	21D101	R		69		LD	HL,MONTHS		MONTH TABLE	
003D	19			70		ADD	HL,DE		FIND MONTH IN TABLE	
003E	5E			71		LD	E,(HL)		DAYS IN MONTH	
003F	2A2702	R		72		LD	HL,(JUL1)		JULIAN SPAN	
0042	AF			73		XOR	A		CLEAR FLAGS	
0043	13			74		INC	DE		ADJUST FOR COMPARE	
0044	ED52			75		SBC	HL,DE		REDUCE COUNT BY MONTH'S LENGTH IN DAYS	
0046	23			76		INC	HL		READJUST REMAINING SPAN	
0047	3011			77		JR	NC,NOTYET		JUMP IF OVER MONTH'S LENGTH	
0049	2A2702	R		78		LD	HL,(JUL1)		JULIAN SPAN	
004C	3A2302	R		79		LD	A,(DAY1)		DAY OF MONTH	
004F	85			80		ADD	A,L		ADD DAY OF MONTH TO SPAN	
0050	6F			81		LD	L,A		MOVE SPAN	
0051	ED52			82		SBC	HL,DE		SEE IF OVER A MONTH	
0053	23			83		INC	HL		READJUST REMAINING SPAN	
0054	382A			84		JR	C,FINISH		LEAVE LOOP IF NOT OVER A MONTH	
0056	AF			85		XOR	A		ZERO ACCU.	
0057	322302	R		86		LD	(DAY1),A		ZERO DAY OF MONTH TO GET OUT OF LOOP	
005A	222702	R		87	NOTYET	LD	(JUL1),HL		SAVE LOOP COUNT	
005D	212202	R		88		LD	HL,MON1		ADDRESS OF MONTHS	
0060	34			89		INC	(HL)		BUMP MONTH	
0061	7E			90		LD	A,(HL)		GET MONTH	
0062	FE00			91		CP	13		13 MONTHS	
0064	38D1			92		JR	C,AGAIN		LOOP ON CARRY	
0066	D60C			93		SUB	12		ADJUST MONTH	
0068	77			94		LD	(HL),A		SAVE MONTH	
0069	212102	R		95		LD	HL,YR1		YEAR ADDRESS	
006C	34			96		INC	(HL)		BUMP YEAR	
006D	7E			97		LD	A,(HL)		GET YEAR	
006E	FE64			98		CP	100		100 YEARS	
0070	38BF			99		JR	C,JULT		LOOP ON CARRY	
0072	D664			100		SUB	100		ADJUST YEAR	
0074	77			101		LD	(HL),A		SAVE YEAR	
0075	212002	R		102		LD	HL,CEN1		ADDRESS OF CENTURY	
0078	34			103		INC	(HL)		BUMP CENTURY	
0079	7E			104		LD	A,(HL)		GET CENTURY	
007A	FE64			105		CP	100		SEE IF ZERO	
007C	28AC			106		JR	Z,ERROR		QUIT ON ERROR	
007E	18B1			107		JR	JULT		LOOP	
				108						
0080	322302	R		109	FINISH	LD	(DAY1),A		SAVE DAY OF MONTH	
0083	6F			110		LD	L,A		MOVE IT	
0084	2600			111		LD	H,0		ZERO MSB	
0086	11D101	R		112		LD	DE,MONTHS		MONTHS TABLE	
0089	3A2202	R		113		LD	A,(MON1)		MONTH	
008C	47			114		LD	B,A		MOVE IT	
008D	1A			115	NEXT1	LD	A,(DE)		DAYS IN MONTH	
008E	85			116		ADD	A,L		ADD UP JULIAN DAYS	

UTILITY PROGRAMS



DATE 12/10/79-15:00 PAGE 3
ASM 5.8

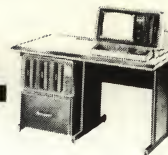
LOC	OBJ CODE	M	STMT	SOURCE	STATEMENT	
008F	3001		117	JR	NC,NEXT2	!JUMP IF NO CARRY
0091	24		118	INC	H	!BUMP MSB
0092	6F		119	NEXT2 LD	L,A	!MOVE LSB
0093	13		120	INC	DE	!BUMP TABLE ADDRESS
0094	10F7		121	DJNZ	NEXT1	!LOOP
0096	222402	R	122	LD	(JULIAN),HL	!SAVE JULIAN DAYS
			123	!		
0099	218A01	R	124	LDBUFF LD	HL,TABLX	!LENGTH TABLE
009C	229301	R	125	LD	(TABDRS),HL	!SAVE TABLE ADDRESS
009F	212002	R	126	LD	HL,CEN1	!ADDRESS OF CENTURY
00A2	6E		127	LD	L,(HL)	!CENTURY
00A3	0E64		128	LD	C,100	!*100
00A5	0600		129	LD	B,0	!MULTIPLY PARAMETER
00A7	60		130	LD	H,0	!ZERO H
00A8	3E17		131	LD	A,17H	!MULTIPLY
00AA	CF		132	RST	8	!SVC
00AB	3A2102	R	133	LD	A,(YR1)	!GET YEAR
00AE	1600		134	LD	D,0	!ZERO MSB
00B0	5F		135	LD	E,A	!MOVE YEAR
00B1	19		136	ADD	HL,DE	!GET TOTAL YEAR
00B2	114006		137	LD	DE,1600	!BASE YEAR 1600
00B5	87		138	OR	A	!ZERO CARRY
00B6	ED52		139	SBC	HL,DE	!YEARS SINCE BASE
00B8	E5		140	PUSH	HL	!SAVE YEARS SINCE BASE
00B9	0601		141	LD	B,1	!DIVIDE PARAM
00BB	0E04		142	LD	C,4	!/4
00BD	3E17		143	LD	A,17H	!DIVIDE
00BF	CF		144	RST	8	!SVC
00C0	23		145	INC	HL	!ADJUST FOR DAY ZERO SAT->MON
00C1	23		146	INC	HL	!SAME
00C2	23		147	INC	HL	!SAME
00C3	23		148	INC	HL	!SAME
00C4	23		149	INC	HL	!SAME
00C5	E5		150	PUSH	HL	!SAVE MOD 4 YEARS
00C6	3A2002	R	151	LD	A,(CEN1)	!GET CENTURY
00C9	D610		152	SUB	16	!BASE CENTURY
00CB	F5		153	PUSH	AF	!SAVE DIFFERENCE
00CC	6F		154	LD	L,A	!MOVE DIFFERENCE
00CD	2600		155	LD	H,0	!ZERO MSB FOR DIVIDE
00CF	0E04		156	LD	C,4	!/4
00D1	3E17		157	LD	A,17H	!DIVIDE
00D3	CF		158	RST	8	!SVC
00D4	F1		159	POP	AF	!DIFFERENCE
00D5	95		160	SUB	L	!SUB # OF QUADRENNIALS
00D6	6F		161	LD	L,A	!MOVE ADJUSTMENT FOR CENTURY
00D7	E5		162	PUSH	HL	!SAVE ADJUSTMENT
00D8	CDDE01	R	163	CALL	LPORNT	!DETERMINE PROPER ADJUSTMENT (LEAP OR NOT)
00DB	E1		164	POP	HL	!ADJUSTMENT
00DC	2001		165	JR	NZ,UPDAT3	!NOT QUADRENNIAL
00DE	2C		166	INC	L	!INC ADJUSTMENT
00DF	EB		167	EX	DE,HL	!ADJUSTMENT TO DE
00E0	E1		168	POP	HL	!MOD 4 YEARS
00E1	AF		169	XOR	A	!ZERO FLAGS
00E2	ED52		170	SBC	HL,DE	!ADJUST MOD 4 YEARS
00E4	EB		171	EX	DE,HL	!MOD 4 YEARS TO DE
00E5	E1		172	POP	HL	!TOTAL YEARS
00E6	19		173	ADD	HL,DE	!TOTAL DAYS OFF FROM BASE TO YEAR IN ?
00E7	ED5B2402	R	174	LD	DE,(JULIAN)	!GET JULIAN DAY



MODEL II TRSDOS

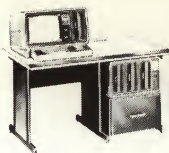
DATM				12/10/79--15:00	PAGE 4
LOC OBJ CODE M STMT SOURCE STATEMENT				ASM 5.8	
00EB	19			ADD HL,DE	!ADD JULIAN DAYS
00EC	0E07			LD C,7	!/7
00EE	0601			LD B,1	!DIVIDE PARAM
00F0	3E17			LD A,17H	!DIVIDE
00F2	CF			RST 8	!SVC
00F3	212602	R	180	LD HL,WK1	!DAY OF WEEK
00F6	71		181	LD (HL),C	!SAVE ABOVE
00F7	0600		182	LD B,0	!DATE PARAMETER
00F9	2A8101	R	183	LD HL,(OUTDSS)	!OUTPUT BUFFER ADDRESS
00FC	3E2D		184	LD A,2DH	!DATE
00FE	CF		185	RST 8	!SVC
00FF	C22A00	R	186	JP NZ,ERROR	!QUIT ON ERROR
0102	219501	R	187	LD HL,NAMDAY	!GET DAY OF WEEK TABLE
0105	79		188	LD A,C	!MOVE DAY OF WEEK
0106	CB07		189	RLC A	!DOUBLE IT
0108	81		190	ADD A,C	!TRIPLE IT
0109	85		191	ADD A,L	!ADD TABLE ADDRESS
010A	6F		192	LD L,A	!FIND TABLE ADDRESS
010B	3001		193	JR NC,DAYOFW	!JUMP IF MSB NC
010D	24		194	INC H	!BUMP MSB ON C
010E	010300		195	LD BC,3	!3 LOOPS
0111	CD8001	R	196	CALL LOADSS	!LD BUFFER
0114	3A2202	R	197	LD A,(MON1)	!GET MONTH
0117	21AA01	R	198	LD HL,NAMONT	!GET TABLE
011A	47		199	LD B,A	!MOVE MONTH
011B	CB07		200	RLC A	!DOUBLE IT
011D	80		201	ADD A,B	!TRIPLE IT
011E	85		202	ADD A,L	!ADD TABLE ADDRESS
011F	6F		203	LD L,A	!FIND TABLE ADDRESS
0120	3001		204	JR NC,MONOFY	!JUMP IF MSB NC
0122	24		205	INC H	!INC MSB ON C
0123	010300		206	LD BC,3	!3 LOOPS
0126	CD8001	R	207	CALL LOADSS	!LOAD BUFFER
0129	3A2302	R	208	LD A,(DAY1)	!GET DAY
012C	CD5D01	R	209	CALL CONSUB	!CONVERT
012F	3A2002	R	210	LD A,(CEN1)	!GET CENTURY
0132	CD5A01	R	211	CALL CONCAR	!CONVERT
0135	3A2102	R	212	LD A,(YR1)	!GET YEAR
0138	CD5A01	R	213	CALL CONCAR	!CONVERT
013B	ED582402	R	214	LD DE,(JULIAN)	!JULIAN DAY
013F	B7		215	OR A	!CLEAR CARRY
0140	CD6101	R	216	CALL CONSUI	!CONVERT AND MOVE
0143	EB		217	EX DE,HL	!OUTPUT BUFFER ADDRESS TO HL
0144	010800		218	LD BC,8	!COUNT FOR MOVE
0147	CD8001	R	219	CALL LOADSS	!BUMP OVER TIME ALREADY IN OUTPUT BUFFER
014A	3A2202	R	220	LD A,(MON1)	!GET MONTH
014D	CD5D01	R	221	CALL CONSUB	!CONVERT
0150	3A2602	R	222	LD A,(WK1)	!DAY OF WEEK
0153	CD5D01	R	223	CALL CONSUB	!CONVERT
0156	AF		224	XOR A	!CLEAR ACCUM
0157	C32D00	R	225	JP ALEXIT	!JUMP TO EXIT
			226	!	
015A	37		227	CONCAR SCF	!SET CARRY FLAG (DON'T CALL EDITM)
015B	1801		228	JR NDCARY	!JUMP OVER CLEARING CARRY FLAG
015D	B7		229	CONSUB OR A	!CLEAR CARRY FLAG (CALL EDITM)
015E	1600		230	NDCARY LD D,0	!ZERO MSB FOR CONVERT
0160	5F		231	LD E,A	!VALUE TO CONVERT
0161	211B02	R	232	CONSUI LD HL,BUFF5	!BUFFER CONVERT ADDRESS

UTILITY PROGRAMS



DATE 12/10/79-15:00 PAGE 5
ASM 5.8

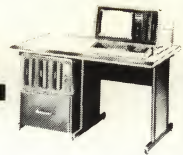
LOC	OBJ CODE	M	STMT	SOURCE	STATEMENT	
0164	0600		233		LD B,0	!CONVERSION PARAMETER
0166	E5		234		PUSH HL	!SAVE ADDRESS
0167	F5		235		PUSH AF	!SAVE FLAGS
0168	3E15		236		LD A,15H	!CONVERT
016A	CF		237		RST B	!SVC
016B	F1		238		POP AF	!GET FLAGS
016C	D49202	R	239		CALL NC,EDIT.M	!EDIT CONVERSION ON NO CARRY
016F	2A9301	R	240		LD HL,(TABDRS)	!TABLE COUNT
0172	4E		241		LD C,(HL)	!LENGTH TO MOVE
0173	23		242		INC HL	!BUMP
0174	229301	R	243		LD (TABDRS),HL	!SAVE ADDRESS
0177	AF		244		XOR A	!ZERO ACCUM.
0178	47		245		LD B,A	!ZERO B
0179	57		246		LD D,A	!ZERO D
017A	3E05		247		LD A,5	!LENGTH OF CONVERSION BUFFER
017C	91		248		SUB C	!DISPLACEMENT IN CONVERSION BUFFER
017D	5F		249		LD E,A	!GET VALUE
017E	E1		250		POP HL	!FRONT OF CONVERSION BUFFER
017F	19		251		ADD HL,DE	!FIND FIRST DIGIT
			252	LOADSS	EQU \$	
0180	110000		253		LD DE,0	!GET BUFFER ADDRESS
			254	OUTDSS	EQU \$-2	
0183	ED80		255		LDIR	!MOVE
0185	ED538101	R	256		LD (OUTDSS),DE	!SAVE ADDRESS
0189	C9		257		RET	
			258	!		
018A	02		259	TABLX	DEFB 2	
018B	02		260		DEFB 2	
018C	02		261		DEFB 2	
018D	03		262		DEFB 3	
018E	02		263		DEFB 2	
018F	02		264		DEFB 2	
0190	02		265		DEFB 2	
0191	02		266		DEFB 2	
0192	01		267		DEFB 1	
			268	!		
0193	8A01	R	269	TABDRS	DEFW TABLX	
			270	!		
0195	4D4F4E54		271	NAMDAY	DEFM '*MONTUEWEDTHUFRISATSUN'	
			272	!		
01AA	2020204A		273	NAMONT	DEFM '* JANFEBMARAPRMAYJUNJUL AUGSEP OCTNOVDEC'	
			274	!		
01D1	00		275	MONTHS	DEFB 00	!NO MONTH
01D2	1F		276		DEFB 31	!JAN
01D3	1C		277	FEB	DEFB 28	!FEB
01D4	1F		278		DEFB 31	!MAR
01D5	1E		279		DEFB 30	!APR
01D6	1F		280		DEFB 31	!MAY
01D7	1E		281		DEFB 30	!JUN
01D8	1F		282		DEFB 31	!JUL
01D9	1F		283		DEFB 31	!AUG
01DA	1E		284		DEFB 30	!SEP
01DB	1F		285		DEFB 31	!OCT
01DC	1E		286		DEFB 30	!NOV
01DD	1F		287		DEFB 31	!DEC
			288	!		
01DE	3A2102	R	289	LPRNT	LD A,(YR1)	!YEAR
01E1	0F		290		LD L,A	!MOVE IT



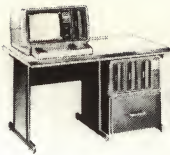
MODEL II TRSDOS

LOC	OBJ	CODE	M	STMT	SOURCE	STATEMENT	DATE	PAGE
01E2	FE00			291	CP	0	12/10/79-15:00	6
01E4	2004			292	JR	NZ,LPI		ASM 5.8
01E6	212002	R		293	LD	HL,CEN1		
01E9	6E			294	LD	L,(HL)		
01EA	2600			295	LD	H,0		
01EC	0601			296	LD	B,1		
01EE	0E04			297	LD	C,4		
01F0	3E17			298	LD	A,17H		
01F2	CF			299	RST	8		
01F3	AF			300	XOR	A		
01F4	B9			301	CP	C		
01F5	C9			302	RET			
				303				
01F6	CDDE01	R		304	FIXFEB	CALL LPORNT		
01F9	3E1C			305	LD	A,28		
01FB	2001			306	JR	NZ,NLTPYR		
01FD	3C			307	INC	A		
01FE	32D301	R		308	NLTPYR	LD (FEB),A		
0201	C9			309	RET			
				310				
0202	0603			311	DTBCVT	LD B,3		
0204	3E30			312	LD	A,'0'		
0206	111B02	R		313	LD	DE,BUFF5		
0209	12			314	LD	(DE),A		
020A	13			315	INC	DE		
020B	10FC			316	DJNZ	LOOP1		
020D	0E02			317	LD	C,2		
020F	EDB0			318	LDIR			
0211	E5			319	PUSH	HL		
0212	211B02	R		320	LD	HL,BUFF5		
0215	04			321	INC	B		
0216	3E15			322	LD	A,15H		
0218	CF			323	RST	8		
0219	E1			324	POP	HL		
021A	C9			325	RET			
				326				
021B				327	BUFF5	DEFS 05		
				328				
0220	00			329	CEN1	DEFB 00		
0221	00			330	YR1	DEFB 00		
0222	00			331	MON1	DEFB 00		
0223	00			332	DAY1	DEFB 00		
0224	0000			333	JULIAN	DEFW 00		
0226	00			334	WK1	DEFB 00		
0227	0000			335	JUL1	DEFW 00		
0229	00			336	CEN2	DEFB 00		
022A	00			337	YR2	DEFB 00		
022B	00			338	MON2	DEFB 00		
022C	00			339	DAY2	DEFB 00		
				340				
022D	00			341	CARRY	DEFB 00		
				342				
022E	CD0202	R		343	EDIT	CALL DTBCVT		
0231	88			344	CP	E		
0232	2851			345	JR	Z,EDITOR		
0234	3E0C			346	LD	A,12		
0236	88			347	CP	E		
0237	384C			348	JR	C,EDITOR		

UTILITY PROGRAMS



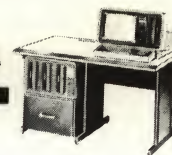
LOC	OBJ CODE	M	STMT	SOURCE	STATEMENT	
<div style="text-align: right;"> DATEM 12/10/79--15:00 PAGE 7 ASM 5.8 </div>						
0239	7B		349		LD A,E	#MONTHS
023A	322202	R	350		LD (MONI),A	#SAVE MONTHS
023D	3E2F		351		LD A,'/'	#DATE DELIMITER
023F	8E		352		CP (HL)	#COMPARE TO STRING
0240	2043		353		JR NZ,EDITOR	#QUIT ON ERROR
0242	23		354		INC HL	#BUMP POINTER
0243	CD0202	R	355		CALL DTBCVT	#CONVERT STRING
0246	203D		356		JR NZ,EDITOR	#QUIT ON ERROR
0248	8B		357		CP E	#CP DAY TO ZERO
0249	283A		358		JR Z,EDITOR	#QUIT ON ERROR
024B	7B		359		LD A,E	#DAY
024C	322302	R	360		LD (DAY1),A	#SAVE DAY
024F	3E2F		361		LD A,'/'	#DATE DELIMITER
0251	8E		362		CP (HL)	#CP TO STRING
0252	2031		363		JR NZ,EDITOR	#QUIT ON ERROR
0254	23		364		INC HL	#BUMP POINTER
0255	CD0202	R	365		CALL DTBCVT	#CONVERT STRING
0258	202B		366		JR NZ,EDITOR	#QUIT ON ERROR
025A	3E0F		367		LD A,15	#15TH CENTURY
025C	8B		368		CP E	#CENTURY
025D	3026		369		JR NC,EDITOR	#QUIT ON ERROR
025F	7B		370		LD A,E	#CENTURY
0260	322002	R	371		LD (CEN1),A	#SAVE CENTURY
0263	CD0202	R	372		CALL DTBCVT	#CONVERT STRING
0266	201D		373		JR NZ,EDITOR	#QUIT ON ERROR
0268	7B		374		LD A,E	#YEAR
0269	322102	R	375		LD (YR1),A	#STORE YEAR
026C	E5		376		PUSH HL	#SAVE POINTER TO INFORMATION
026D	CD0601	R	377		CALL FIXFEB	#FIX FEB.
0270	3A2202	R	378		LD A,(MON1)	#MONTH
0273	21D101	R	379		LD HL,MONTHS	#MONTHS TABLE
0276	85		380		ADD A,L	#ADJUST TABLE ADDRESS
0277	6F		381		LD L,A	#MOVE LSB OF MONTH ADDRESS
0278	3001		382		JR NC,OVFLOW	#JUMP ON NO OVERFLOW
027A	24		383		INC H	#MSB FOR OVERFLOW
027B	3A2302	R	384	OVFLOW	LD A,(DAY1)	#DAY
027E	3D		385		DEC A	#ADJUST FOR COMPARE
027F	8E		386		CP (HL)	#COMPARE DAY
0280	E1		387		POP HL	#RESTORE POINTER
0281	3002		388		JR NC,EDITOR	#QUIT ON ERROR
0283	AF		389		XOR A	#CLEAR FLAGS
0284	C9		390		RET	
0285	3E03		391	EDITOR	LD A,3	#ERROR ON CALL
0287	B7		392		OR A	#SET FLAG
0288	C9		393		RET	
			394		#	
0289	0604		395	LOADR	LD B,4	#LOOP COUNT
028B	212002	R	396		LD HL,CEN1	#FRONT OF FIRST DATE INFO.
028E	112902	R	397		LD DE,CEN2	#FRONT OF SECOND DATE INFO.
0291	C9		398		RET	#RETURN TO SENDER
			399		#	
0292	0E05		400	EDITM	LD C,5	#SIGN. DIGITS COUNT
0294	0604		401		LD B,4	#MAX SPACES TO INSERT
0296	3E30		402		LD A,'0'	#LEADING ZERO VALUE
0298	8E		403	EDITM1	CP (HL)	#ASC-II ZERO ?
0299	C0		404		RET NZ	#LEAVE IF NOT
029A	0D		405		DEC C	#DEC SIGN. DIGIT COUNT
029B	3620		406		LD (HL),' '	#PUT IN LEADING SPACE



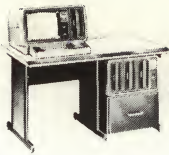
MODEL II TRSDOS

LOC	OBJ CODE	M	STMT	SOURCE STATEMENT	DATE	12/10/79--15:00	PAGE 8	ASM 5.8
029D	23		407	INC HL				!NEXT BYTE
029E	10F8		408	DJNZ EDITM1				!LOOP
02A0	C9		409	RET				!RETURN
02A1	CD2E02	R	410	! TWODAT				
02A4	C22A00	R	411	CALL EDIT				!EDIT INPUT INFORMATION
02A7	3E20		412	JP NZ,ERROR				!QUIT ON ERROR
02A9	BE		413	LD A,(HL)				!BLANK
02AA	C22A00	R	414	CP				!COMPARE WITH DELIMITER IN STRING
02AD	23		415	JP NZ,ERROR				!QUIT ON ERROR
02AE	E5		416	INC HL				!BUMP INFORMATION POINTER
02AF	CD8902	R	417	PUSH HL				!SAVE POINTER
02B2	0T0400	R	418	CALL LOADR				!LOAD REGISTERS WITH FRONT OF DATE INFO.
02B5	EDB0		419	LD BC,4				!LOOP COUNT
02B7	E1		420	LDIR				!MOVE
02B8	CD2E02	R	421	POP HL				!POINTER TO INFORMATION
02B8	C22A00	R	422	CALL EDIT				!EDIT INFORMATION
02BE	210000		423	JP NZ,ERROR				!QUIT ON ERROR
02C1	222702	R	424	LD HL,00				!ZERO HL
02C4	212D02	R	425	LD (JUL1),HL				!ZERO JULIAN COUNT
02C7	3600		426	LD HL,CARRY				!CARRY ACCUMULATOR
02C9	CD8902	R	427	LD (HL),0				!ZERO IT
02CC	1A		428	CALL LOADR				!LOAD REGISTERS WITH FRONT OF DATE INFO.
02CD	4E		429	LD A,(DE)				!GET FIRST DATE
02CE	77		430	LD C,(HL)				!MOVE 1ST DATE
02CF	79		431	LD (HL),A				!
02D0	12		432	LD A,C				!
02D1	23		433	LD (DE),A				!
02D2	13		434	INC HL				!BUMP POINTER
02D3	10F7		435	INC DE				!BUMP POINTER
02D5	CD8902	R	436	DJNZ SWITCH				!LOOP
02D8	1A		437	CALL LOADR				!LOAD REGISTERS WITH FRONT OF DATE INFO.
02D9	BE		438	LD A,(DE)				!2ND DATE
02DA	23		439	CP (HL)				!1ST DATE
02DB	13		440	INC HL				!BUMP
02DC	DA2A00	R	441	INC DE				!BUMP
02DF	2002		442	JP C,ERROR				!QUIT ON ERROR
02E1	10F5		443	JR NZ,GOOD				!QUIT LOOKING IF DDD2 IS SMALLER DATE
			444	DJNZ LOW1ST				!LOOP
			445	!				
02E3	CD8902	R	446	CALL LOADR				!LOAD REGISTERS WITH FRONT OF DATE INFO.
02E6	05		447	DEC B				!REDUCE COUNT BY ONE
02E7	1A		448	LD A,(DE)				!GET 2ND DATE
02E8	BE		449	CP (HL)				!CP TO NEW DATE (FIRST DATE)
02E9	23		450	INC HL				!BUMP
02EA	13		451	INC DE				!BUMP
02EB	203E		452	JR NZ,COMPAR				!ADD A MONTH AND UPDATE
02ED	10F8		453	DJNZ EVENUP				!LOOP UNTIL UPDATED
			454	!				
02EF	3A2302	R	455	LD A,(DAY1)				!EARLIER DAY
02F2	5F		456	LD E,A				!MOVE IT
02F3	AF		457	XOR A				!ZERO ACCU.
02F4	57		458	LD D,A				!MOVE IT
02F5	2A2702	R	459	LD HL,(JUL1)				!TOTAL JULIAN DAYS
02F8	ED52		460	SBC HL,DE				!SUBTRACT DAYS
02FA	3006		461	JR NC,OVERIT				!SKIP IF NO CARRY
02FC	E5		462	PUSH HL				!SAVE TOTAL
02FD	212D02	R	463	LD HL,CARRY				!CARRY ACCUMULATOR
0300	35		464	DEC (HL)				!ADJUST ACCORDING TO CARRY

UTILITY PROGRAMS



LOC	OBJ CODE	M	STMT	SOURCE	STATEMENT	DATE	12/10/79--15:00	PAGE	9
						ASM	5.8		
0301	E1		465		POP HL				#TOTAL
0302	3A2C02	R	466	OVERIT	LD A,(DAY2)				#THE OTHER DAY
0305	5F		467		LD E,A				#MOVE IT
0306	T9		468		ADD HL,DE				#ADD DAYS IN MONTH TO JULIAN TOTAL
0307	EB		469		EX DE,HL				#TOTAL TO DE
0308	3E00		470		LD A,0				#ZERO ACCU. (KEEP FLAGS)
030A	212D02	R	471		LD HL,CARRY				#CARRY ACCUMULATOR
030D	3001		472		JR NC,ONOUT				#GO ON IF NO CARRY
030F	34		473		INC (HL)				#BUMP CARRY ACCUMULATOR
0310	BE		474	ONOUT	CP (HL)				#SEE IF ANY CARRYS ARE LEFT
0311	C22A00	R	475		JP NZ,ERROR				#QUIT ON ERROR
0314	211B02	R	476		LD HL,BUFF5				#CONVERSION BUFFER
0317	0600		477		LD B,0				#CONVERSION PARAMETER
0319	E5		478		PUSH HL				#SAVE FRONT OF BUFFER
031A	3E15		479		LD A,15H				#CONVERT
031C	CF		480		RST 8				#SVC
031D	CD9202	R	481		CALL EDITM				#EDIT CONVERSION
0320	E1		482		POP HL				#FRONT OF BUFFER
0321	010500		483		LD BC,5				#COUNT
0324	CD8001	R	484		CALL LOADSS				#MOVE TO OUTPUT BUFFER
0327	AF		485		XOR A				#GOOD INDICATION
0328	C32D00	R	486		JP ALEXIT				#END
			487						
032B	CDF601	R	488	COMPAR	CALL FIXFEB				#FIX FEB.
032E	3A2202	R	489		LD A,(MON1)				#MONTH
0331	1600		490	MORE	LD D,0				#ZERO MSB
0333	5F		491		LD E,A				#MOVE MONTH
0334	21D101	R	492		LD HL,MONTHS				#MONTHS TABLE
0337	19		493		ADD HL,DE				#FIND MONTH
0338	5E		494		LD E,(HL)				#GET # OF DAYS IN MONTH
0339	2A2702	R	495		LD HL,(JUL1)				#TOTAL FOR JULIAN DAYS
033C	19		496		ADD HL,DE				#ADD UP TOTAL
033D	3006		497		JR NC,FLOW				#JUMP IF NO OVERFLOW
033F	E5		498		PUSH HL				#SAVE TOTAL
0340	212D02	R	499		LD HL,CARRY				#CARRY ACCUMULATOR
0343	34		500		INC (HL)				#BUMP CARRY
0344	E1		501		POP HL				#TOTAL
0345	222702	R	502	FLOW	LD (JUL1),HL				#SAVE TOTAL
0348	212202	R	503		LD HL,MON1				#MONTH
034B	34		504		INC (HL)				#BUMP MONTH
034C	7E		505		LD A,(HL)				#MONTH TO ACCU.
034D	FE00		506		CP 13				#13 MONTHS
034F	3892		507		JR C,GOOD				#LOOP ON CARRY
0351	D60C		508		SUB 12				#READJUST
0353	77		509		LD (HL),A				#STORE IT
0354	212102	R	510		LD HL,YR1				#YEAR
0357	34		511		INC (HL)				#BUMP YEAR
0358	7E		512		LD A,(HL)				#YEAR TO ACCU.
0359	FE64		513		CP 100				#100 YEARS
035B	3886		514		JR C,GOOD				#LOOP ON CARRY
035D	D664		515		SUB 100				#READJUST
035F	77		516		LD (HL),A				#STORE IT
0360	212002	R	517		LD HL,CEN1				#CENTURY
0363	34		518		INC (HL)				#BUMP CENTURY
0364	C3E302	R	519		JP GOOD				#LOOP



MODEL II TRSDOS

CROSS REFERENCE
SYMBOL VAL M DEFN REFS

DATM

12/10/79--15:00

PAGE 10

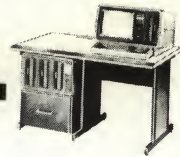
AGAIN	0037	R	67	92																
ALEXIT	002D	R	60	225	486															
BUFF5	021B	R	327	232	313	320	476													
CARRY	022D	R	341	426	463	471	499													
CEN1	0220	R	329	102	126	151	210	293	371	396	517									
CEN2	0229	R	336	397																
COMPAR	032B	R	488	452																
CONCAR	015A	R	227	211	213															
CONSUI	0161	R	232	216																
CONSUB	015D	R	229	209	221	223														
DATM	0000	R	34																	
DAY1	0223	R	332	79	86	109	208	360	384	455										
DAY2	022C	R	339	466																
DAYOFW	010E	R	195	193																
DTBCVT	0202	R	311	343	355	365	372													
EDIT	022E	R	343	48	411	422														
EDITM	0292	R	400	239	481															
EDITM1	0298	R	403	408																
EDITOR	0285	R	391	345	348	353	356	358	363	366	369	373	388							
ERROR	002A	R	58	40	43	49	52	106	186	412	415	423	442							
				475																
EVENUP	02E7	R	448	453																
FEB	01D3	R	277	308																
FINISH	0080	R	109	84																
FIXFEB	01F6	R	304	65	377	488														
FLOW	0345	R	502	497																
GOOD	02E3	R	446	443	507	514	519													
JUL1	0227	R	335	56	72	78	87	425	459	495	502									
JULDAT	0016	R	48																	
JULIAN	0224	R	333	122	174	214														
JUL1	0031	R	65	57	99	107														
LDBUFF	0099	R	124																	
LOADR	0289	R	395	418	428	437	446													
LOADSS	0180	R	252	196	207	219	484													
LOOP1	0209	R	314	316																
LQW1ST	02D8	R	438	444																
LP1	01EA	R	295	292																
LPORNT	01DE	R	289	163	304															
MON1	0222	R	331	66	88	113	197	220	350	378	489	503								
MON2	022B	R	338																	
MONOFY	0123	R	206	204																
MONTHS	01D1	R	275	69	112	379	492													
MORE	0331	R	490																	
NAMDAY	0195	R	271	187																
NAMONT	01AA	R	273	198																
NEXT1	008D	R	115	121																
NEXT2	0092	R	119	117																
NOCARY	015E	R	230	228																
NOTYET	005A	R	87	77																
NILPYR	01FE	R	308	306																
ONDUT	0310	R	474	472																
OUTDSS	0181	R	254	44	183	256														
OVERIT	0302	R	466	461																
OVFLOW	027B	R	384	382																
SWITCH	02CC	R	429	436																
TABDRS	0193	R	269	125	240	243														
TABLX	018A	R	259	124	269															
TWDDA1	02AT	R	411	47																

PAGE 11

3/25

17-11-1917





HERZ50

Set Up for 50 Hz AC power (non-USA users)

DO HERZ50

starts the utility to change the system for 50 Hz operation. HERZ50 is a DO-file.

This utility is provided for customers in areas where the AC power is 50 rather than 60 Hz. It should not be used by any other customers.

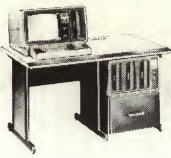
HERZ50 is a DO-file that makes a few changes in the video display software of TRSDOS. Only the drive zero diskette is changed. Be sure it is write-enabled before you start the DO-file. Once the HERZ50 changes are done, they will remain in effect for that diskette.

To perform the change, type under TRSDOS READY:

DO HERZ50

The program provides several opportunities for you to abort the process by pressing **BREAK**. Pressing any other key during a pause will cause the program to continue.

Once the change has been made, you will need to reset the system to put the change into effect. This loads the new video display software into RAM.



PATCH

Change the contents of a disk file

PATCH *programfile* *A=address, F=findstring, C=changestring*

This is the form to use when you are patching a program stored with the "P" (program) attribute (see DIR). Files created with DUMP will fall into this category.

programfile specifies the file you want to change. If it is a system file, no password is necessary. If it is a protected user file, the password must be included.

A=aaaa

aaaa is the starting address of the data to be changed. This is where the data resides in memory when the program is loaded. *aaaa* is a four-digit hexadecimal value without the 'X' notation.

F=findstring specifies the string that is currently in the patch area.

C=changestring specifies what data is to replace *findstring*. *changestring* must contain the same number of bytes as does *findstring*.

Both *findstring* and *changestring* can be in hexadecimal or ASCII form. In hexadecimal form, each byte to be changed is represented as a two-digit hexadecimal value. In ASCII form, each byte to be changed is represented by the ASCII character corresponding to that byte value. ASCII strings must be enclosed by single or double quotes.

PATCH *datafile* *R=record, B=startingbyte, F=findstring, C=changestring*

This is the form to use when you are patching a data file, i.e., a file stored with the "D" attribute (see DIR). BASIC programs and data files fall into this category.

datafile specifies the file you want to change. If it is a system file, no password is necessary. If it is a protected user file, the password must be included.

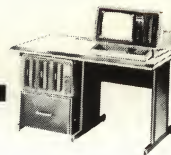
R=record

record tells which record contains the data to be changed, and is a decimal number from 1 to 65536.

B=startingbyte

startingbyte specifies the position of the first byte to be changed. It is a decimal number from 1 to 256.

F=findstring and **C=changestring** are described above



This utility lets you make minor corrections in any disk file, provided that:

1. You know the existing contents and location of the data you want to change.
2. You want to replace one string of code or data with another string of the same length.

You can use PATCH to make minor changes to your own machine-language programs; you won't have to change the source code, re-assemble it, and re-create the file. You can also use it to make minor replacement changes in data files.

Another application for PATCH is to allow you to implement any modifications to TRSDOS that may be supplied by Radio Shack. That way, you do not have to wait for a later release of the operating system.

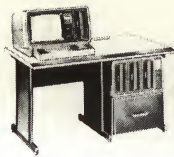
Note: If you press **BREAK** during a patch operation, before any changes have been made in the file, PATCH will close the file and return you to TRSDOS. The file will be unchanged. Once PATCH has begun changing the file, pressing **BREAK** will have no effect.

Using PATCH on a TRSDOS System File

When Radio Shack releases a modification to TRSDOS, you will receive a printout of the exact PATCH commands that are required to perform the change.

To implement such a change, you would follow these steps:

1. Make a backup copy of the diskette to be patched.
2. Insert the TRSDOS disk to be changed into one of the drives. The diskette must be write-enabled.
3. In the TRSDOS READY mode, type in the specified PATCH command.
4. After completion of the patch, test the diskette in drive zero to see that it is operational as a TRSDOS system diskette. You may have to reset the Computer.



Using PATCH on a Program File

Remember that in this context, "program files" refers strictly to those files stored with the "P" attribute. Use the DIR command to find out the attributes of a file. BASIC programs have the "D", not the "P", attribute. (See instructions for changing data files.) Program files are created with DUMP.

Suppose you want to change seven bytes in a machine-language program file. First determine where the seven-byte sequence resides in RAM when the program is loaded. Then make sure that your replacement string is the same length as that of the original string. For example, you might write down the information as follows:

File to be changed: VDREAD

Start address: X'5280'

Sequence of code to be changed: X'CD2C25E5'

Replacement code: X'000000C9'

Then you could use the following command:

```
PATCH VDREAD A=5280, F=CD2C25E5, C=000000C9
```

Using PATCH on a data file (including BASIC programs)

If the file is stored with the "D" attribute, you specify the patch area in terms of the logical record which contains the data, and the starting byte of the data in that record. (The TRSDOS LIST command gives this information.)

For example, suppose in a file called NAMEFILE you need to change a 12-byte sequence. When you LIST the file, you find that the sequence is located in record 128, and that the sequence starts at byte 14. Write down the information like this:

File to be changed: NAMEFILE

Record number: 128

Starting byte: 14

Sequence of text to be changed: "JOHN'S DINER"

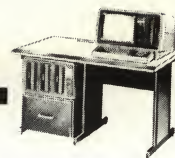
Replacement text: "JACK'S PLACE"

Then use the following command:

```
PATCH NAMEFILE R=128, B=14, F="JOHN'S DINER", C="JACK'S PLACE"
```

Notice that either string can include a single-quote, as long as the string is surrounded by double-quotes. If you wanted to include a double-quote inside either string, you would have to enclose that string in single-quotes.

Note: The string you are changing must be wholly contained inside the specified record. If it spans two records, you will have to perform the patch operation twice, once for each record.



Error Conditions

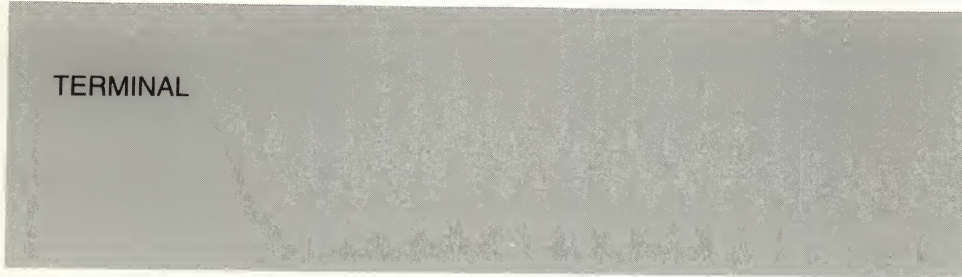
If a TRSDOS error occurs during the patch operation, you will receive the appropriate ****ERROR nn**** message, and the patch will be terminated without changing the file.

PATCH can also produce the following messages:

PATCH STRING TOO LONG—ABORT	This occurs when you are patching a data file and the patch string spans two records. You will need to perform the patch in two steps, one for each record that contains a part of the string to be changed.
FILE CONTAINS VARIABLE-LENGTH RECORDS—ABORT	You can only patch fixed length record files.
STRING NOT FOUND	The find-string string was not found at the patch location you specified. Before patching a file, you must know the exact patch location and the existing contents of that location.
ADDRESS OUT OF PROGRAM-LOAD RANGE—ABORT	This occurs when you attempt to patch a program file, and some or all of the patch string is outside the RAM area where the program resides when it is loaded. Check the A=aaaa parameter. Also be sure that the <i>findstring</i> and <i>changestring</i> aren't longer than you intended for them to be.



TERMINAL



This is a versatile program designed to allow communications between the Model II and another computer running a host program. **TERMINAL** is designed primarily for transmission and reception of ASCII text rather than machine-language object code.

Input/output is through serial channel A. In most applications, hookup will be through telephone lines via a modem.

TERMINAL has three modes of operation:

- **Menu** — Allows you to select or change options, even execute TRSDOS library commands
- **Interactive terminal**—Transmits your keyboard input and displays incoming data
- **Transmit from RAM**—For high-speed transfer of prepared data. Incoming data is displayed on the screen.

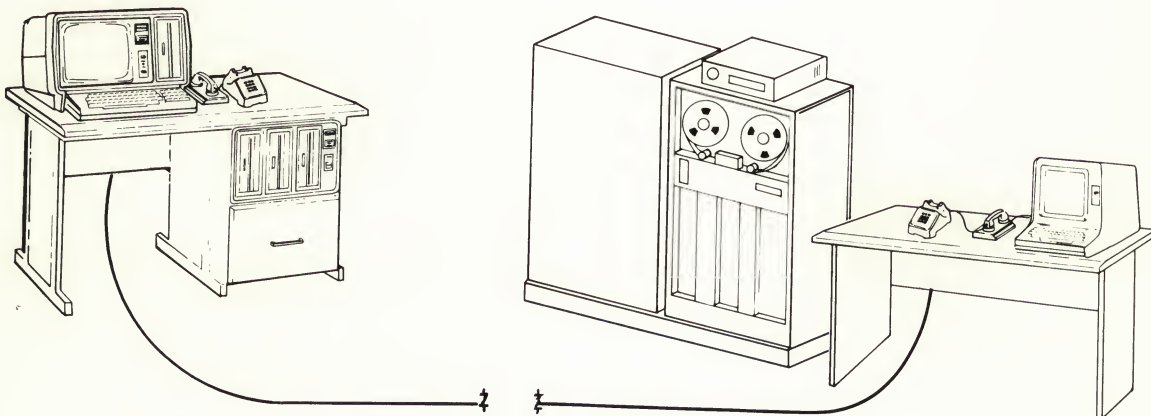
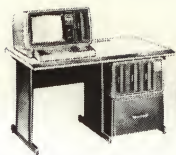


Figure 1. A typical terminal/host configuration.



Setting Up

For communications through ordinary telephone lines, you will need a modem such as the Radio Shack Telephone Interface II, Catalog Number 26-1171, and the Model II RS-232 Cable, 26-4403.

1. Set up the modem according to its instructions, and connect it to channel A on the back panel of the Model II display console. Unless channel B is connected to another device, you must install the serial terminator on that channel.
2. Find out what RS-232-C parameters are required by the host program you are going to use:
 - Baud rate
 - Word length
 - Parity
 - Number of stop bitsYou will need to initialize channel A accordingly (see "Running Terminal").
3. Set the modem to originate or answer mode — whichever is *different* from the mode used by the host program you are going to communicate with. Also set it to full or half duplex, again depending on the requirements of the host program.
4. Turn on the modem and the Model II computer system.

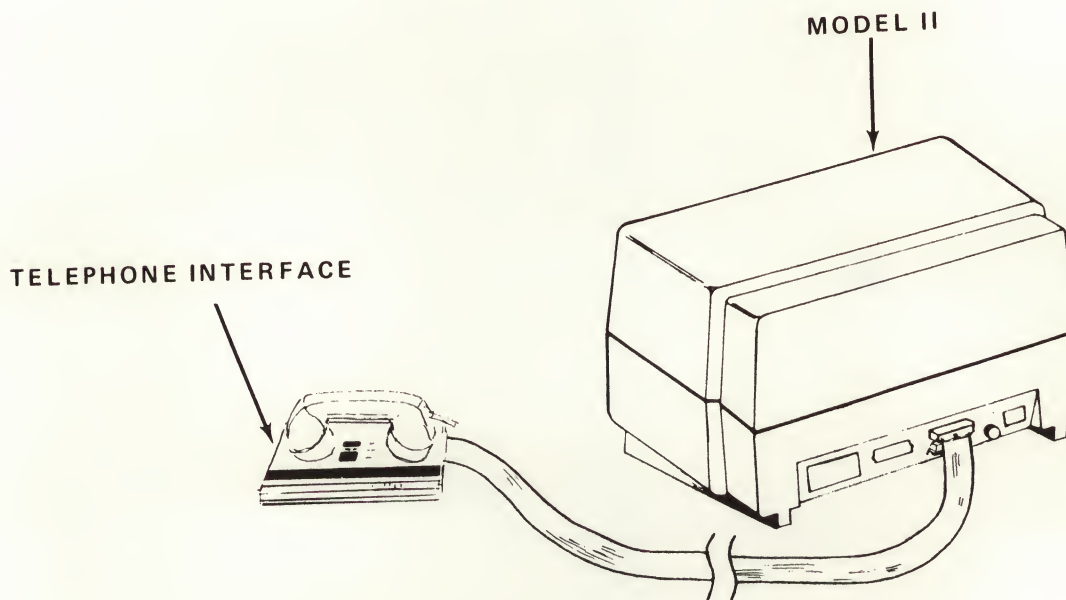
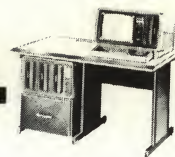


Figure 2. Connection of Model II to a modem.



Running Terminal

Since **TERMINAL** allows you to enter any **TRSDOS** library command, it is not necessary to initialize channel A or the printer before starting.

From the **TRSDOS** **READY** mode, you can start **TERMINAL** by typing:

```
TERMINAL
```

The program starts up in the menu mode, with the prompt:

```
-- ENTER MENU SELECTION ..
```

Note: When we show computer prompts and user input in the same example, we highlight the user input with a gray background.

Now is a good time to initialize channel A according to the requirements of the host program you are going to communicate with. Type:

```
-- ENTER MENU SELECTION S
```

The program will prompt you to type in a **TRSDOS** command. Type in the **SETCOM** command just as you would in the **TRSDOS** **READY** mode. For example,

```
ENTER TRSDOS COMMAND (1-79)  
SETCOM A=(300,7,N,2) ENTER
```

would enable channel A with 300 baud, seven-bit words, no parity and two stop bits. After executing the command, control will return to **TERMINAL**'s menu mode.

If you plan to use the printer option of **TERMINAL** (described later on), you should also initialize the printer now, with the **FORMS** command. Type:

```
-- ENTER MENU SELECTION S ENTER
```

and enter the appropriate **FORMS** command.

To select another menu command, type in the letter specified in the menu table. For example, type:

```
-- ENTER MENU SELECTION M ENTER
```

to redisplay the entire menu.



Modes of Operation

Menu Mode

This is an off-line mode, i.e., you cannot transmit characters to the host program, and if characters are sent to you, they will be lost. This is the only mode in which you can select menu options described later on. From it, you can also enter the transmit from RAM or interactive terminal mode.

Interactive Terminal Mode

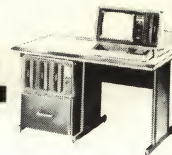
You can enter this mode from the menu with the T command; you also enter this mode automatically after completion of an auto sign-on or a transmission from the RAM buffer.

In the interactive terminal mode, characters you type are sent to the host program, and incoming characters are displayed as they are received. If the host program echoes your transmissions, they too will appear on the display; if not, you can select the echo option and TERMINAL will display your keyboard input.

Incoming characters can be saved in the RAM buffer (R option) and can be output to the printer (P option).

If transmission errors occur, TERMINAL will display a descriptive error message and wait for the error condition to be corrected. When it is, normal I/O will resume in the interactive terminal mode.

To return to the menu mode, press **HOLD** .





Transmit from RAM (and Auto Sign-On)

You enter this mode via the X command. The contents of the RAM buffer are sent to the host program, and control passes to the interactive mode. Auto sign-on (O command) works just like transmit from RAM; the following comments apply to both operations.

The RAM buffer contains prepared text which you have typically loaded from a disk file with the G option. (If you are using auto sign-on, your auto sign on message is sent.) You can send the data one line at a time when the host program prompts you that it is ready (W option), or send it in a continuous stream.

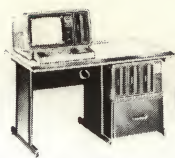
During the transmission, incoming text will be displayed on the screen. If the host program echoes your transmissions, you will be able to verify that the data was sent accurately.

During the transmissions, you can adjust the delay between characters by repeatedly pressing the  (faster) and  (slower) keys. If echoed data appears garbled, slow down the transmissions. If not, you might want to speed it up.

If a break character is received in this mode, TERMINAL will pause until the next character is received. If a X'13' is received, TERMINAL will pause until a X'11' is received. (By convention, X'13' is called the DC3 signal and means pause; X'11' is called the DC1 signal and means resume.)

If transmission errors occur, TERMINAL will display a descriptive error message and wait for the error condition to be corrected. When it is, normal I/O will resume.

To exit from this mode at any time, press **HOLD** . You will be returned to the menu.



Details of the Menu

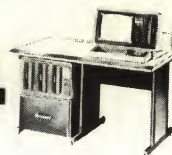
M **Display Menu**

After you have entered several menu commands, the menu begins scrolling off the display. Use the M command to clear the display and redisplay the menu.

S **Perform System Command**

Whenever you need to perform a TRSDOS library command, use this command as shown previously. After execution of a TRSDOS library command, control will return to TERMINAL's menu.

Note: You cannot execute FORMAT, BACKUP, BASIC, or a single-drive, multi-diskette COPY from TERMINAL.



B Set/Change Break Character Key

This command lets you define which code will be recognized by **TERMINAL** as a break character. It also lets you define a key which will send that code.

Any code from zero to 255 can be specified. When such a character is received from the host program, **TERMINAL** will warn you so you can take appropriate action. If it is in the transmit from RAM mode, it will pause the transmission until the next character is received.

Note: If **TERMINAL** receives a break sequence (extended null character) from the host program, it will handle it like a break character.

This option also lets you specify which key on the Model II keyboard will send the break code. Any key except **BREAK** or **CTRL C** can be used.

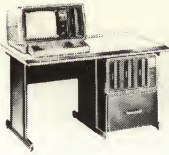
For example, suppose the host program uses X'10' as a break code — meaning it wants you to stop sending data. Then you would want to modify **TERMINAL**'s break code by typing:

```
-- ENTER MENU SELECTION B ENTER
CURRENT BREAK CHARACTER IS 03 HEX, CURRENT KEY IS 1B HEX
CHANGE? (Y/N) Y ENTER
ENTER NEW CHAR VALUE IN HEX (2) 10 ENTER
ENTER NEW KEY (1) ESC

CURRENT BREAK CHARACTER IS 10 HEX, CURRENT KEY IS 1B HEX
--ENTER MENU SELECTION ..
```

W Set/Change Prompt Wait Character

This command affects **TERMINAL** operation in the transmit from RAM mode and during auto sign-on. It does not affect operation in the interactive terminal mode.



MODEL II TRSDOS

The prompt wait feature allows you to use the high-speed transmit from RAM mode — even when the host program can only accept one line at a time. Typically, the host program sends you a prompt such as a question mark or colon when it is ready for the next line. In the interactive keyboard mode, you would simply wait until this prompt is displayed; the prompt wait feature makes TERMINAL do the same thing while in the transmit from RAM mode.

When the feature is on, TERMINAL will send one line at a time, and wait for a prompt character from the host program before sending each line. (A line is defined as a string of characters terminated by a carriage return X'0D'.)

You can define the prompt wait character as any keyboard character from X'20' to X'7F'.

Leave the prompt wait feature off when the host program is simply storing characters as received, and is not sending a ready-for-next-line prompt. TERMINAL will transmit text from RAM in a continuous stream.

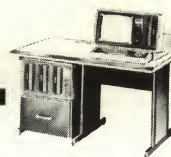
Suppose the host program sends a question-mark (code X'3F') when it is ready for a line of text. Then type:

```
--ENTER MENU SELECTION W  ENTER
PROMPT OPTION NOW OFF
PRESS ENTER TO LEAVE AS-IS OR ENTER NEW CHARACTER ?
PROMPT OPTION ON WITH '?' AS THE CHARACTER
--ENTER MENU SELECTION ..
```

From now on during auto sign-on or transmit from RAM, TERMINAL will wait for the question-mark prompt before it sends a line.

Note: When you start the transmit from RAM (X option) or auto sign-on (O option), the first line will be sent immediately, without waiting for a prompt. Each subsequent line will be sent after a prompt has been received.

To turn the prompt wait feature off, press **HOLD** when the program asks for a new character.



F Set/Change F1 & F2 Keys

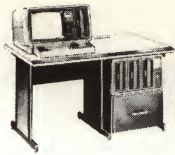
This command lets you program F1 and F2 to output any code from zero to 255. This is useful when you will be using a particular code frequently.

For example, suppose the host program accepts a X'13' as a pause control, and X'11' as a resume. After receiving a pause character, it waits for a resume character before sending any more text.

Although you can send these codes from the keyboard (CTRL Q is a X'11' and CTRL S is a X'13'), it would be more convenient to program F1 and F2 to send these codes. Type:

```
-- ENTER MENU SELECTION F ENTER
F1 KEY WILL SEND A 01 HEX CODE
CHANGE? (Y/N) Y ENTER
ENTER NEW CHAR VALUE IN HEX (2) 11
F1 KEY WILL SEND A 11 HEX CODE
F2 KEY WILL SEND A 02 HEX CODE
CHANGE? (Y/N) Y ENTER
ENTER NEW CHAR VALUE IN HEX (2) 13
F2 KEY WILL SEND A 13 HEX CODE
-- ENTER MENU SELECTION ..
```

Now when you type **F1** in the interactive terminal mode, TERMINAL will transmit the resume control X'11'; for **F2**, the pause control X'13'.



L Toggle Line Feed Option

This command tells TERMINAL how to handle an incoming line feed X'0A'. When the option is on, all line feeds are ignored. When it is off, they are not ignored.

The option is useful if the host program always sends a line feed after a carriage return. Since the TRSDOS display and printer drivers automatically perform a line feed after a carriage return is sent, the incoming line feed is redundant. Therefore the line feed option should normally be on.

To toggle (change the state of) the line feed option, type:

--- ENTER MENU SELECTION **L** **ENTER**

The new state of the option (on or off) will be displayed, and the menu prompt will return.

P Toggle Printer Option

This command turns the printer option on and off. When the option is on, incoming text will be copied to the printer as it is received and displayed. Whenever you use D command while this option is on, the RAM buffer text will be copied to the printer.

Be sure you have initialized the printer with the FORMS command before attempting to use it.

To toggle the printer option, type:

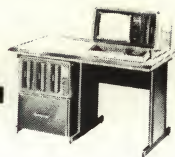
--- ENTER MENU SELECTION **P** **ENTER**

The new state of the option (on or off) will be displayed, and the menu prompt will return.

TERMINAL uses a circular buffer for efficient output to the printer. However, if characters come in too fast, they will not be printed. They *will* be displayed, though, and will be saved in RAM if the buffer is open. (Check your printer's specifications for maximum character input rate. At 300 baud, 7-bit characters may come in as fast as 30 per second.)

Another way to output to the printer is via the D (display RAM buffer) command. When the printer option is on, the D command copies the RAM buffer contents to the printer.

To minimize hookup time, don't use the printer option while on-line with the host program. Save the incoming text in RAM instead. After completion of the hookup, turn the printer option on and use the D command to get a hard copy of the data.



E Toggle Self Echo Option

Some host programs echo the text you send. That is, as the host receives each character, it sends it right back to you. In this case, what you send will be displayed on the screen. When communicating with this type of host program, set your modem to full duplex.

If the host program does not echo your text, then what you send (keyboard input or text from RAM) will not be displayed. In this case, you should use the self-echo option, which displays everything you type or transmit from RAM. With such host programs, set your modem to half duplex.

To toggle the echo option, type:

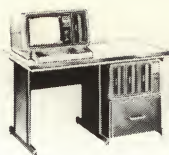
--- ENTER MENU SELECTION **E** **ENTER**

The new state of the option (on or off) will be displayed, and the menu prompt will return.

R Toggle RAM Buffer Option

This command pertains to the interactive terminal mode only. It lets you save in RAM some or all the data that is received, by "opening" and "closing" the RAM buffer. That way, you can examine the data later with the D command, or save the data in a disk file with the C command.

Whenever you open the RAM buffer, you have a choice of resetting it or retaining its current contents. In the latter case, new incoming text will be loaded after the existing text in the RAM buffer.



MODEL II BASIC

To toggle the RAM buffer option, type:

--- ENTER MENU SELECTION R **ENTER**

The new state of the option (on or off) will be displayed. If you have just opened the buffer, you will receive the following prompt:

```
RAM BUFFER NOW OPEN
RESET RAM BUFFER? (Y/N) . .
```

If you type Y **ENTER**, the buffer will be reset and previous contents will be lost. For further information, see "Using the RAM Buffer."

A Build Auto Sign-On Message

This command lets you prepare an automatic sign-on to be sent to the host program with the O option. Typically, the message will contain responses to the standard sign-on questions provided when you first call up a host program.

The message can be up to 60 characters, consisting of any characters that can be entered from the keyboard. The message can include control characters. To embed a carriage return (X'0D') in the message, press **↓**. It will be echoed as ±, but a carriage return will be stored.

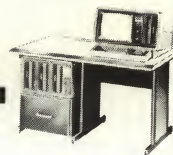
If you enter any other control character in the message, it will also be displayed as a ±, but the true control code will be sent. (When you display a message, control codes will not be shown at all.)

For example, suppose the host terminal requires responses to the following prompts during sign-on:

```
USER ID?
USER PASSWORD?
PROGRAM NAME?
```

Instead of typing in information each time you call up the host program, you can store the responses in an auto sign-on buffer. In this case, the buffer might contain the following information:

```
FTW-779 <X'0D'>
LUMMOX  <X'0D'>
MENU    <X'0D'>
```

To set up this auto sign-on message, type:

```
-- ENTER MENU SELECTION A ENTER  
THE CURRENT AUTO-SIGN ON IS  
  
CHANGE? (Y/N) Y ENTER  
ENTER AUTO SIGN-ON MESSAGE (1-60)  
FTW ↓ LUMMOX ↓ MENU ENTER  
THE CURRENT AUTO SIGN-ON IS  
FTW  
LUMMOX  
MENU  
--ENTER MENU SELECTION ..
```

The blank line above indicates that the original auto sign-on was blank or contained non-display characters.

G Get Disk File into RAM Buffer

This command lets you load text stored in a disk file into the RAM buffer. Then you can send it to the host program via the transmit from RAM command. The previous contents of the RAM buffer are lost.

The disk file can contain fixed- or variable-length records, and fixed length records can have any length. However, only ASCII files should be loaded and sent. You can send BASIC programs as long as you saved them with the A (ASCII) option.

For example, suppose you have created a document stored in the file DOCUMENT.TXT. You want to send it to the host program. To get the file into the RAM buffer, type:

```
-- ENTER MENU SELECTION G ENTER  
ENTER FILESPEC (1-34)  
DOCUMENT.TXT ENTER
```

TERMINAL will load the file and return to the menu. The RAM buffer will be closed.

If the host program is ready to accept data, you can now send it with the X command. After transmission is complete, TERMINAL will go to the interactive terminal mode.



C Copy RAM Buffer to Disk

This command creates a disk file copy of the text in the RAM buffer. The new file will have a record length of one. Use this command to save data that has been received into the RAM buffer in the interactive terminal mode. To minimize hookup time, you will probably want to do this after ending the connection to the host program. Or if the RAM buffer is full, save it in a disk file, then reset it and re-open it to accept more data.

For example, suppose you have just received a report in the interactive terminal mode, and you want to save it in a disk file named REPORT. Type:

```
-- ENTER MENU OPTION C  ENTER  
ENTER FILESPEC (1-34)  
REPORT  ENTER
```

The new file will be created (if REPORT already exists, it will be overwritten with the new data), and the RAM buffer contents and status will be unchanged.

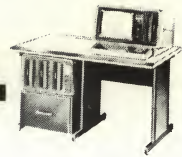
To stop the copy process, press **HOLD**. The disk file will be closed and you will be returned to the menu.

D Display RAM Buffer

This command displays the contents of the RAM buffer. To pause the display, press **F1**. To continue, press **F2**. If the printer option is on when you issue this command, the text will also be output to the printer. To enter the command, type:

```
-- ENTER MENU SELECTION D  ENTER
```

To stop the display function, press **HOLD**. You will be returned to the menu.



X Transmit RAM Buffer and Enter Term Mode

This option puts you in the transmit from RAM mode, in which the current contents of the RAM buffer are sent to the host program. When the entire buffer has been sent, TERMINAL goes into the interactive terminal mode. For details see "Transmitting from RAM."

To stop transmitting from RAM, press **HOLD** . You will be returned to the menu.

O Enter Terminal Mode with Auto Sign-On

This command starts transmission of the current auto sign-on message. After the message is sent, TERMINAL enters the interactive terminal mode. For details, see "Transmitting from RAM."

To stop transmitting the auto sign-on, press **HOLD** . You will be returned to the menu.

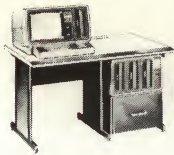
Note: Most host programs cannot receive anything until the host program has sent the first prompting message. Because of this, you should:

1. Go to the interactive terminal mode (T option) when connection is first made, and wait for the host to send its first prompt character.
2. Press **HOLD** to return to the menu.
3. Start the auto sign-on (O option).

T Enter Terminal Mode

This command puts you directly into the interactive terminal mode. While in this mode, press **HOLD** to return to the menu.

For details, see "Interactive Terminal."



Using the RAM Buffer

The RAM buffer is used to store incoming text (R option) and prepared text from a disk file (G option) so that it can be sent rapidly. The RAM buffer helps reduce costly hookup time, by letting you perform time-consuming operations — preparing data or printing it out — while you are off-line.

Size

For 32K Model II systems, the buffer can contain up to 12K (12,288) bytes of text; for 64K systems, 44K (45,056).

For 32K systems using 300 baud (i.e., 30 characters per second), it will take approximately 7 minutes to fill the buffer in the interactive terminal mode; for 64K systems, 25 minutes.

If the buffer becomes filled during a load from disk (G command) or while receiving data in the interactive terminal mode, a warning message will be displayed and the buffer will be closed. If you are loading a disk file, you will be returned to the menu and the buffer will contain the data that was loaded. If you are in the interactive terminal mode, normal I/O will continue, except that it will no longer be saved in the buffer.

Saving the RAM Buffer

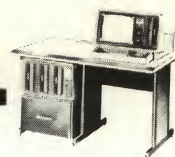
When the buffer is filled in the interactive mode (or when you suspect it will be soon), do the following:

1. Transmit a pause or break control character to the host program.
2. Return to the menu by pressing **HOLD**.
3. Copy the RAM buffer to a disk file (C command).
4. Reset the RAM buffer (R command).
5. Return to the interactive terminal mode (T command).

Opening and Closing the RAM Buffer

Often during interactive I/O, you want to save only portions of the text. The R command lets you do this. Each time you are about ready to receive some important data, do the following:

1. Transmit a pause or break control character to the host program.
 2. Return to the menu by pressing **HOLD**.
 3. Toggle the RAM buffer status. If it is not off, toggle it again. If it is on, you have the option of resetting it or leaving it as is. To add new data onto the end of old, *do not* reset it. To delete old data, *do* reset it. For details and examples, see R command later on.
 4. Return to the interactive terminal mode (T command).
 5. Direct the host program to resume transmission. The data will now be saved in the RAM buffer as it is received.
-



Saving the Options You Have Selected

You can create a customized version of `TERMINAL` — one which starts up with the options you have selected. For example, the break character and key, `F1` and `F2` characters and auto sign-on message, could all be saved so you won't have to set them each time you start the program.

Options which may be saved in a customized program:

- Prompt wait and definition of prompting character
- Definition of break character from host program and assignment of a break key on your computer
- `F1` and `F2` characters
- Line feed option
- Printer option
- Self-echo option
- Auto sign-on option
- Receive into RAM option (not recommended)
- Speed of transmit from RAM and auto sign-on (as set by the `↑` and `↓` keys).
Once you find out the maximum rate of transmission the host program can handle, you'll probably want that to become the default rate.

After selecting the options for your customized version, you use the `DUMP` command to create a new program file. Terminal resides from `X'3000'` to `X'3FFF'`, and its entry point is `X'3000'`.

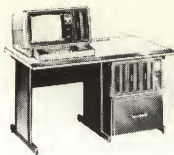
You must give this customized program a name other than `TERMINAL` — and leave `TERMINAL` in its original configuration. Suppose you want to call your customized version `MINE`. Then type:

```
-- ENTER MENU SELECTION S  ENTER
ENTER TRSDOS COMMAND (1-79)
DUMP MINE START=3000, END=3FFF  ENTER
```

Now when you type:

```
TRSDOS READY
MINE  ENTER
```

your customized version of `TERMINAL` will start.



Sample Uses

To Send a BASIC Program

The program must be stored in an ASCII-format disk file. For example, suppose you are in BASIC and you are ready to save a program on drive 1 with the file name SORTDATA. Then type:

```
Ready  
>SAVE "SORTDATA:1", A ENTER  
Ready  
>SYSTEM ENTER  
TRSDOS READY  
TERMINAL ENTER
```

Once you have set up the modem and initialized channel A as explained previously, call up the host program and place the telephone handset into the modem. The modem's ready light should come on, indicating that you are receiving the carrier signal from the host program.

```
-- ENTER MENU SELECTION T ENTER
```

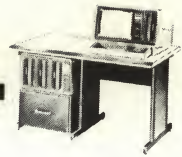
Now go through the necessary sign-on with the program. When you want to send the BASIC program, press **HOLD** to return to the menu. (If you want to use the prompt wait option, select it now.) Then type:

```
-- ENTER MENU SELECTION G ENTER  
ENTER FILESPEC (1-34)  
SORTDATA:1 ENTER
```

Terminal will load the program into RAM. Make sure the host is ready to receive the program, then type:

```
-- ENTER MENU SELECTION X ENTER
```

and the program will be sent to the host. Press **HOLD** if you want to stop the transmission for any reason. You will return to the menu. Otherwise you will go into the interactive terminal mode when the program has been sent.



To Receive a BASIC Program

Suppose you are communicating with the host program and it is ready to send you an ASCII-format BASIC program. Before telling the host to go ahead, first go to the menu and type as follows:

```
-- ENTER MENU SELECTION R ENTER
```

If the buffer is now closed, repeat the R command and **TERMINAL** will display the message:

```
RAM BUFFER NOW OPEN  
RESET RAM BUFFER (Y/N) Y ENTER
```

This opens and clears the buffer. Now go back to the interactive terminal mode (T option) and tell the host to send the program.

After the entire program has been received, press **HOLD** to return to the menu; then type:

```
-- ENTER MENU SELECTION C ENTER  
ENTER FILESPEC (1-34)  
NEWPROG ENTER
```

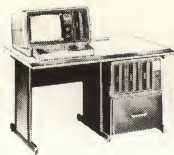
This will copy the BASIC program in RAM into a disk file (we named this one NEWPROG).

To try out the program, exit **TERMINAL** and go into **BASIC**. Then type:

```
Ready  
>LOAD "NEWPROG" ENTER
```

After the program has loaded, you should examine it to see if it needs modification to run under Model II BASIC. Pay particular attention to BASIC statements and functions involving input/output:

- Keyboard
- Video Display
- Line Printer
- Disk Files



MODEL II TRSDOS

Error Conditions

In the interactive terminal mode, transmit from RAM mode, or during auto-sign on, TERMINAL may detect errors related to the serial transmission. In such cases, it will display an error message in reverse video (black on white); if possible, it will continue normal I/O.

Here are the messages that may be produced while you are in the interactive terminal mode:

P	Parity error. The received character will be displayed after the reverse P.
O	Over-run. At least one character has been received but not picked up by TERMINAL. This will happen if you are in the menu mode while the host program is sending characters.
F	Framing error. The received character will be displayed after the reverse F. Check your SETCOM parameters to see that they match the requirements of the host program.

The following messages can be produced in any mode except the menu:

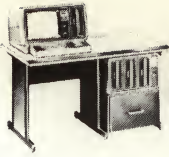
DATA CARRIER LOST DATA CARRIER RESTORED	Check the telephone/modem connection. TERMINAL will pause until the carrier is restored. If TERMINAL was transmitting from RAM or sending an auto sign-on, it will start over at the beginning of the text when the data carrier is restored.
BREAK SEQUENCE RECEIVED	If the host program sends a break sequence, or sends TERMINAL's own break character, this message will be displayed; if TERMINAL is in the transmit from RAM or auto sign-on mode, it will pause until the next character is received from the host.



TERMINAL: A Quick Summary



Feature/ Option	Menu	Transmit From RAM & Auto Sign-On	Interactive Terminal
Return To Menu With HOLD	N/A	Yes	Yes
Execute TRSDOS Commands	Use "S" Command	No	No
Break Character & Key	Use "B" Option	When Break Sequence Or Break Is Received, Pauses Transmission until next Character Is Received	Transmits Character From Keyboard
Wait For Prompt	"W" Option	Yes	No
Program F1 And F2	"F" Option	No	Yes
Ignore Line Feeds After Carriage Returns	"L" Option	Yes	Yes
Output To Printer	"P" Option	Yes	Yes
Self-Echo	Use "E" Option	Yes	Yes
Receive Into RAM	"R" Option	No	Yes
Auto Sign-On	Set Up' With "A" Option; Send With "O" Command	Active Here	Enters This Mode When Done

Continued on next page



MODEL II TRSDOS

TERMINAL A Quick Summary, continued

Feature/ Option	Menu	Transmit From RAM & Auto Sign-On	Interactive Terminal
Get Disk File Into Ram	"G" Command	No	No
Copy Ram To Disk	"C" Option	No	No
Transmit Ram Buffer	"X" Option	Active Here	Enters This Mode When Done
Recognize DC1/DC3 Resume/Pause	No	Yes	No
Display Ram Buffer	"D" Command	No	No
Adjust Speed Of Transmit	No	Yes, With  And 	No
Redisplay Menu	"M" Command	No	No
Save Customized Program	"S" Command Dump X'3000' X'3FFF'	No	No



XFERSYS

Transfer Operating System

XFERSYS

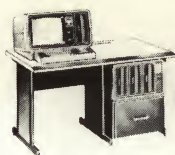
This utility lets you upgrade a TRSDOS diskette to a later version of the operating system. It uses the built-in drive, thus simplifying the upgrade process for one-drive customers. (Customers with multi-drive systems may find it more convenient to use the COPY command to put user files onto the latest TRSDOS diskette, rather than using XFERSYS.)

XFERSYS copies all system files from a new release "master diskette" onto an old release "user diskette." System files which already exist on the user diskette are replaced by those from the master diskette. Files which do not exist on the user diskette are added. *User files (programs and data) are unaffected.*

We suggest you always use the new-release diskette supplied by Radio Shack (or a duplicate created with BACKUP) as your master. Using one of your XFERSYS-updated duplicates as a master will work, but could produce diskettes with a slower processing speed. This is because the system files are in their optimum position on the new-release diskette from Radio Shack. This is not necessarily true of diskettes created with the XFERSYS utility.

Special Note for Users of Radio Shack Applications Programs

Certain Radio Shack applications programs (e.g., Inventory Control) use most or all available space on the diskette. Even a "minimal system" containing no utilities except XFERSYS may not fit on such applications diskettes. In this case, do not update that diskette. Continue using the applications program under 1.1. Radio Shack will continue to accept customer questions regarding applications programs released under 1.1.



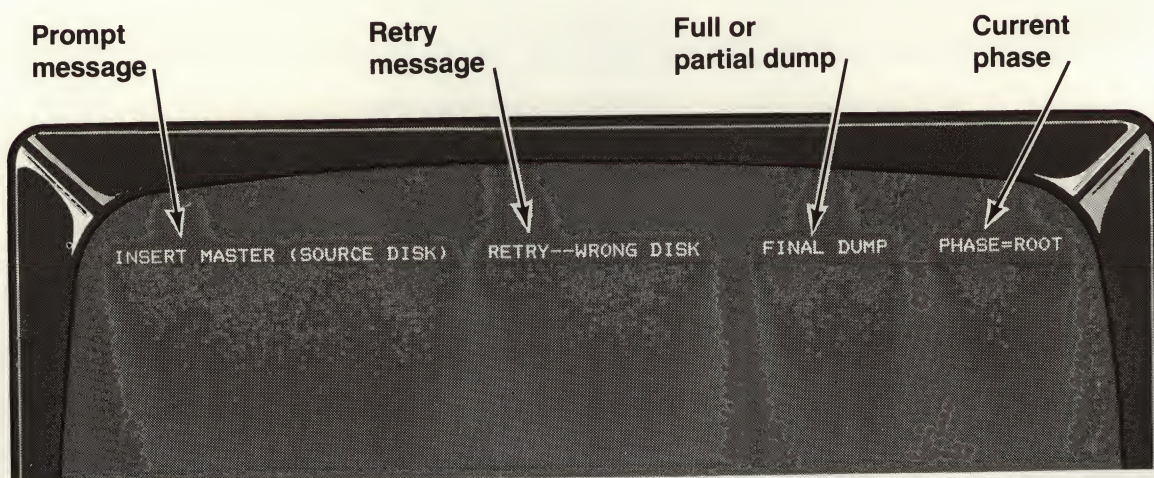
MODEL II TRSDOS

Steps to upgrade a diskette

1. Make backup copies of all diskettes to be upgraded. This is an important precautionary step. These backup copies should be kept until the upgrading process is complete and confirmed.
2. Leave both diskettes (master and user) write-enabled during the upgrading process (write-protect notch covered).
3. Insert the new release of TRSDOS into the built-in drive (drive zero) and reset the Computer. Then type:

XFERSYS

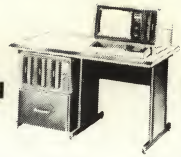
4. During the upgrading process, the program will prompt you alternately to insert the user and master diskettes into drive zero. Additional information will appear on the same line as the prompting message. For example:



Prompt message: MASTER DISK refers to the new release diskette; USER DISK refers to the diskette you are upgrading.

Retry message: If you insert the wrong diskette, the re-try message will be displayed; insert the correct diskette and the program will continue.

Note: The TRSDOS version of the user diskette must not be later than that of the master diskette. This prevents you from accidentally "downgrading" from a later to an earlier version.



Full or partial dump: This tells you whether the current phase of the operation requires one master/user diskette swap (FULL) or multiple diskette swaps (PARTIAL) for completion.

Current phase: This indicates the type of files that are being copied (ROOT or SECONDARY). This information is provided solely as positive feedback that the process is continuing; you do not need it for any other reason.

5. When the process has been completed, the message:

END OF XFERSYS CONVERSION — RE-BOOT

Reset the computer. The user diskette is now upgraded to the same version as that of the master diskette.

Error Conditions

Under certain conditions, the program will display a TRSDOS error message:

* * ERROR *nn* * *

followed by:

XFER CONVERSION — ABORT REBOOT

This indicates that an input/output error occurred during the upgrading process. Reset the computer and handle the error as you would under TRSDOS, then start over.

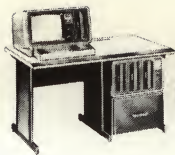
Special Note Regarding DISK FULL message

A DISK FULL message (ERROR 27) means that there is not enough room on the user diskette to contain the new files on the master diskette. The user diskette will be unusable in this case, requiring you to use your backup copy.

Before proceeding, you should make a backup copy of this diskette. You will then need to kill some files from the diskette to free up space for the new operating system.

Or you may choose to kill some non-essential files from your new release of TRSDOS. (Be sure you first BACKUP the original!). See the TRSDOS 1.2 Update General Instructions for details.

Once you have acquired more space on the backup user diskette (or prepared a "minimal system" diskette for use as a master), start over with XFERSYS.



MODEL II TRSDOS

Other messages that may occur are:

PROCESS/PROCEDURE ERROR

This indicates a procedural or internal computational error occurred. If DEBUG is on, you must turn it off before starting XFERSYS. Reset the computer and start over.

DATA ERROR

This indicates that data in the system files has been incorrectly modified before XFERSYS was started. Retry the process. If the error recurs, you will not be able to upgrade this particular diskette.

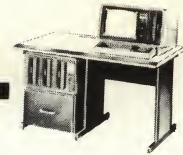
XFERSYS LOCKED – RE-BOOT

This indicates an illegal attempt to use XFERSYS. Follow steps 1 through 4 above.

Section 4

Technical Information

This chapter explains TRSDOS on a technical level. You do not need it to use the Operator Commands, nor do you need it to run BASIC applications programs on the Computer. You do need it to write assembly programs which use System routines. You may also find the information incidentally useful in programming with BASIC.



Diskette Organization

Model II uses single-sided, double-density diskettes. Each diskette contains 77 tracks, numbered 0-76.

Each track contains 26 sectors, numbered 1-26. Each sector contains 256 bytes, except for track 0 sectors, which contain 128 bytes. The total capacity of a diskette is:

$$(76 * 26 * 256) + (1 * 26 * 128) = 509,184 \text{ bytes}$$

Disk Space Available to User

Sector 26 of each track is reserved for System use, giving the user 25 sectors per track. On System diskettes, 65 tracks are available for the user; on non-System diskettes, 75 tracks are available.

Details: Track 0 is reserved by the System. Another track (usually track 44) is reserved by the System for the diskette directory. On Operating System diskettes, ten additional tracks are used for System files.

Unit of Allocation

The only unit of disk space allocation is the "granule". A TRSDOS granule is defined as 5 sectors. Therefore the smallest non-empty file consists of 5 sectors; i.e., one granule.

NON-SYSTEM DISKETTE	TRACKS	GRANULES	SECTORS	BYTES
1	75	375	1875	480,000
—	1	5	25	6400
—	—	1	5	1280
—	—	—	1	256

Table 4.1 Space Available to User



Disk Files

Methods of File Allocation

Model II provides two ways to allocate disk space for files: Dynamic Allocation and Pre-Allocation.

Dynamic Allocation

With Dynamic Allocation, the System allocates granules only at the time of write. For example, when a file is first Opened for output, no space is allocated. The first space allocation is done at the first write. Additional space is added as required by subsequent writes.

With dynamically allocated files, unused granules are de-allocated (recovered) when the file is Closed.

Pre-Allocation

With Pre-Allocation, the file is allocated a specified number of granules when it is Created. Pre-Allocated files can only be created by the operator command CREATE.

TRSDOS will dynamically extend (enlarge) a Pre-Allocated file as needed for subsequent write operation. However, TRSDOS will not de-allocate unused granules when a pre-allocated file is Closed. The way to reduce the size of a Pre-Allocated file is to Copy it to a dynamically allocated file and Kill the Pre-Allocated one.

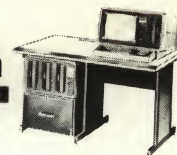
Record Length

The Model II transfers data to and from diskettes one sector at a time; i.e., in 256-byte blocks. These are the System's "physical" records.

User records or "logical" records are the buffers of data you wish to transfer to or from a file. These can be from 1 to 256 bytes long.

The Operating System will automatically "block" your logical records into physical records which will be transferred to disk, and "de-block" the physical records into logical records which are used by your program. Therefore your **only** concern during file-access is with logical records. You never need to worry about physical records, sectors, tracks, etc. This is to your benefit, since physical record lengths and features may change in later TRSDOS versions, while the concept of logical records will not.

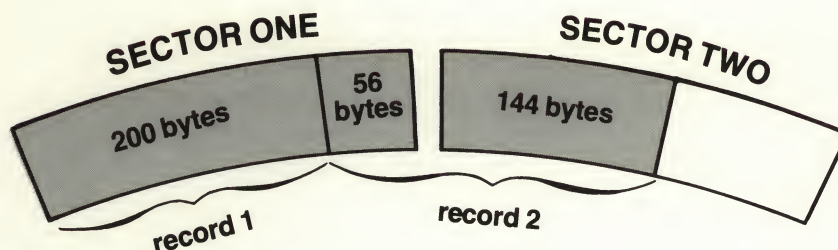
From this point on, the term "record" refers to a "logical record".



Spanning

If the record length is not an even divisor of 256, the records will automatically be spanned across sectors.

For example, if the record length is 200, Sectors 1 and 2 will look like this:



Fixed-Length and Variable Length Records

Model II files can have either fixed-length or variable-length records. Files with fixed-length records will be referred to as FLRs; files with variable length records, VLRs.

Record length in an FLR file is set when the file is Opened for the first time. This length can be any value from 1 to 256 bytes. Once set, the record length in an FLR cannot be changed, unless the file is being over-written with new data.

Record length in a VLR file is specified in a one-byte length-field at the beginning of each record. The record-lengths in a VLR file can vary. For example, the first record in a file might have a length of 32; the second, 17; the third, 250; etc.

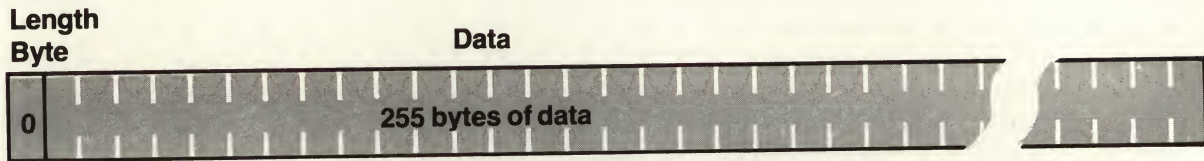
The record-length byte indicates the entire length of the record, **including** the length-byte. This can be any value from 0 to 255. A value of 1 can be used, but it has no meaning.



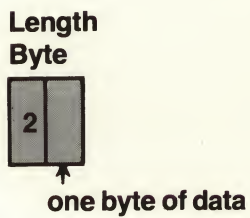
Model II TRSDOS

Examples:

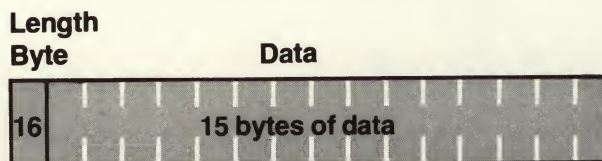
A length-byte value of zero indicates that the record contains 255 bytes of data:

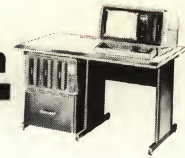


A length-byte value of 2 indicates that the record contains 1 byte of data:



A length-byte value of 16 indicates that the record contains 15 bytes of data:





Record Processing Capabilities

Model II TRSDOS allows both Direct and Sequential file access. Direct access—sometimes called “random access”, but “direct” is more descriptive—allows you to process any record you specify.

Note: A file can contain up to 65535 records. Records are numbered from 0 (beginning of file) to 65534. A record number of 65535 indicates the end of file (EOF). These limits will be changed in a later release of TRSDOS.

Sequential access allows you to process records in sequence: Record N, N+1, N+2, . . . With sequential access, you do not specify a record number; instead, the Operating System accesses the next record following the last record processed.

For files with fixed length records (FLRs) you can position the current record pointer to the beginning of the file, end of file, or to any record in the file. In short, you can use Direct and/or Sequential Access with FLRs at any time during processing.

For files with variable length records (VLRs) you can only position to the beginning of the file or to the end of file. You **cannot** position to any other record in the file, since the position of interior VLRs cannot be calculated. If short, you can only use Sequential access with VLRs.

The Direct access routines are Direct-Read and Direct-Write; the Sequential access routines are Read-Next and Write-Next. Direct access routines always access the record you specify. Sequential access routines always access the record **following** the last record processed. (When the file is first opened, sequential processing starts with record 0.)

Examples

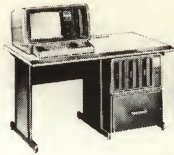
Assume you have a Fixed Length Record file currently Open. Here are some typical sequences you can accomplish via the file processing routines.

1. Read and/or write records in the file — in any order

This is done using Direct-Write and Direct-Read routines. You could read record 5, write at end of file, read record 3, write record 3, etc.

2. Sequential Read (or Write) beginning anywhere in the file.

First you would do a Direct-Read to the record where you want to start reading or writing. After that, you would do sequential reads or writes until done.



Model II TRSDOS

3. Sequential Write starting at end of file.

First do a Direct-Write to the end of the file. Then do sequential writes until done.

4. Determine the number of records in a file.

First do a Direct-Read to end of file, then use the LOCATE routine to get the current record number, which now equals *number of records + 1*.

Examples with Variable Length Records

Here are examples of ways to read or write to VLR files:

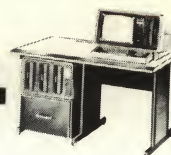
1. Start reading or writing sequentially at first record

Open the file and start reading or writing sequentially until done.

2. Sequential Write starting at end of file

(same process as Example 3 above).

Note: Whenever you write to a VLR, the end of the file is **automatically** reset to the last record you write. This means you cannot update a VLR file directly; you must read in the file and output the updated information to a new VLR file.



How to Use the Supervisor Calls

Supervisor Calls (SVCs) are Operating System routines available to any user program. The routines alter certain System functions and conditions; provide file access; perform I/O to the Keyboard, Video Display, and Printer; and perform various computations.

All the SVCs leave memory above X'2FFF' untouched. Only those Z-80 registers used to pass parameters from the SVC are altered. All others are unaffected. However, all the prime registers are used by the System; they are not restored.

Each SVC is assigned a Function Code. These codes run from 0 through 127. Only the first 96 are defined by the System; codes 96-127 are available for user definition.

To specify a given Supervisor Call, your program refers to the SVC's Function Code.

Calling Procedure

All SVCs are accomplished via the RST 8 instruction.

1. Load the Function Code for the desired SVC into the A register. Also load any other registers which are needed by the SVC, as detailed under "Supervisor Calls."
2. Execute a RST 8 instruction.
3. Upon return from the SVC, the Z flag will be set if the function was successful. If the Z flag is not set, there was an error. The A register contains the appropriate error code (except after certain computational SVCs, which use the A register to return other information).

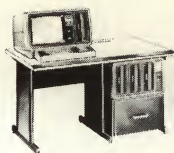
Examples

Time-Delay

```
LD    BC,TIMCNT           ; LENGTH OF DELAY
LD    A,6                 ; FUNCTION CODE 6 = DELAY-SVC
RST   B                   ; JUMP TO SVC
;    DELAY OVER-PROGRAM CONTINUES HERE
```

Output a line to the Video Display

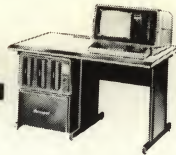
```
LD    HL,MSG              ; POINT TO THE MESSAGE
LD    B,10                ; B=CHARACTER COUNT
LD    C,0DH              ; C=CTRL CHAR. TO ADD AT END
LD    A,9                 ; CODE 9 = DISPLAY LINE-SVC
RST   B                   ; JUMP TO SVC
JR    NZ,GOTERR           ; JUMP IF I/O ERROR
;    IF NO ERROR THEN PROGRAM CONTINUES HERE
```



Model II TRSDOS

Get a character from the Keyboard

```
GETCHAR LD    A,4           ; CODE 4 = GET CHARACTER-SVC
        RST    B           ; JUMP TO SVC
        JR     NZ,GETCHAR   ; DO AGAIN IF NO CHARACTER
;   CHARACTER IS IN REGISTER B
```

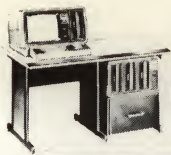



Error Codes and Messages

Register A usually contains a return code after any function call. The Z flag is set when no error occurred. Exceptions are certain computational routines, which use the A and F registers to pass back data and status information.

CODE	MESSAGE
0	NO ERROR FOUND
1	BAD FUNCTION CODE ON SVC CALL OR NO FUNCTION EXISTS
2	CHARACTER NOT AVAILABLE
3	PARAMETER ERROR ON CALL
4	CRC ERROR DURING DISK I/O OPERATION
5	DISK SECTOR NOT FOUND
6	ATTEMPT TO OPEN A FILE WHICH HAS BEEN CLOSED.
7	DRIVE DOOR WAS OPENED WHILE FILE OPEN FOR WRITE
8	DISK DRIVE NOT READY
9	INVALID DATA PROVIDED BY CALLER
10	MAXIMUM OF 16 FILES MAY BE OPEN AT ONCE
11	FILE ALREADY IN DIRECTORY
12	NO DRIVE AVAILABLE FOR AN OPEN
13	WRITE ATTEMPT TO A READ ONLY FILE
14	WRITE FAULT ON DISK I/O
15	DISK IS WRITE PROTECTED
16	DCB IS MODIFIED AND IS UNUSABLE
17	DIRECTORY READ ERROR
18	DIRECTORY WRITE ERROR
19	IMPROPER FILE NAME (file spec)
20	FAD READ ERROR
21	FAD WRITE ERROR
22	FID READ ERROR
23	FID WRITE ERROR
24	FILE NOT FOUND
25	FILE ACCESS DENIED DUE TO PASSWORD PROTECTION
26	DIRECTORY SPACE FULL
27	DISK SPACE FULL

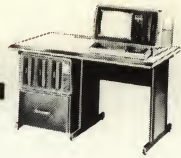
(Continued on next page)



Model II TRSDOS

28	ATTEMPT TO READ PAST EOF
29	READ ATTEMPT OUTSIDE OF FILE LIMITS
30	NO MORE EXTENTS AVAILABLE (16 MAXIMUM)
31	PROGRAM NOT FOUND
32	UNKNOWN DRIVE NUMBER (filespec)
33	DISK SPACE ALLOCATION CANNOT BE MADE DUE TO FRAGMENTATION OF SPACE
34	ATTEMPT TO USE A NON PROGRAM FILE AS A PROGRAM
35	MEMORY FAULT DURING PROGRAM LOAD
36	PARAMETER FOR OPEN IS INCORRECT
37	OPEN ATTEMPT FOR A FILE ALREADY OPEN
38	I/O ATTEMPT TO AN UNOPEN FILE
39	ILLEGAL I/O ATTEMPT
40	SEEK ERROR
41	DATA LOST DURING DISK I/O (HARDWARE FAULT)
42	PRINTER NOT READY
43	PRINTER OUT OF PAPER
44	PRINTER FAULT (MAY BE TURNED OFF)
45	PRINTER NOT AVAILABLE
46	NOT APPLICABLE TO VLR TYPE FILES
47	REQUIRED COMMAND PARAMETER NOT FOUND
48	INCORRECT COMMAND PARAMETER
49	HARDWARE FAULT DURING DISK I/O
50	**UNKNOWN ERROR CODE**

Table 4.2 Error Codes



Supervisor Calls

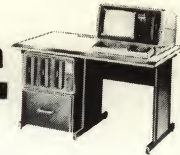
In this section we will use the following notation:

NOTATION	MEANING
RP = data	The register-pair RP contains the data.
$n1 < R < n2$	The register R contains a value greater than n1 and less than n2
(RP) = data	The register-pair RP contains the address of ("points to") the data.
NZ = Error	If Z flag is not set, an error occurred.

When a range is not given, any representable number can be used. For example, a register can contain any value from 0-255.

The contents of this section are:

System Control Supervisor Calls	4/15
Keyboard Supervisor Calls	4/27
Video Display Supervisor Calls	4/31
Line Printer Supervisor Calls	4/45
File Access Supervisor Calls	4/49
Computational Supervisor Calls	4/61
Serial Communications Supervisor Calls	4/75

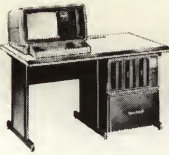


System Control

Supervisor calls described in this section:

FUNCTION CODE	NAME	PURPOSE
0	INITIO	Initializes all I/O drivers
2	SETUSR	Sets up a user-defined SVC
3	SETBRK	Sets up BREAK key processing program
15	DISKID	Reads a diskette ID
25	TIMER	Set timer to interrupt program
36	JP2DOS	Returns to TRSDOS (TRSDOS READY)
37	DOSCMD	Sends TRSDOS a command and then returns to TRSDOS READY
38	RETCMD	Sends TRSDOS a command and return to caller
39	ERROR	Displays "ERROR <i>number</i> "
52	ERRMSG	Returns Error Message to Buffer

Table 4.3. System Control Supervisor Calls



INITIO **Initialize I/O (function code 0)**

This routine initializes all input/output drivers. It calls all of the other initialization routines. There are no parameters.

Note: This routine has been done already by the System. Users never need to call it, except in extreme error conditions.

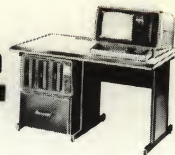
Error Conditions

A = 0

Exit Conditions

NZ = Error

A = Error Code



SETUSR

Set User (function code 2)

This routine sets or removes a user vector. This gives you the ability to add SVC functions. Function codes 96-127 are available for user definition.

Once added, such a function can then be called via the RST 8 instruction, just like the System's SVC routines.

Your routine must reside above X'27FF', and should end with a RET instruction.

To change a previously defined function, you must first remove the old vector.

Entry Conditions

- HL = Entry address of your routine (when C is not equal to 0)
- B = Function code to be used, $95 < \text{code} < 128$
- C = Set/Reset code. If C=0, remove the vector. Otherwise, add the vector.
- A = 2

Exit Conditions

- HL = Removed vector address (when C=0 on entry)



SETBRK

Set **BREAK** (function code 3)

This routine lets you enable the **BREAK** key by defining a **BREAK** -key processing program. Whenever the **BREAK** key is pressed, your processing program takes over. On entry to the **BREAK** processing program, the return address of the interrupted routine is on the top of the stack and can be returned to with a RETurn instruction. All Z-80 register contents are preserved upon entry to the **BREAK** program.

The routine also lets you disable the **BREAK** key, by removing the address of the processing program. While **BREAK** key is enabled, you cannot change processing programs; you must disable it first.

The **BREAK** key processing program must reside above X'27FF'.

See "Handling Programmed Interrupts" for programming information.

Entry Conditions

HL = Address of **BREAK** key processing program. When **BREAK** is pressed, control transfers to this address.

If HL = 0, then address of previous processing program is removed.

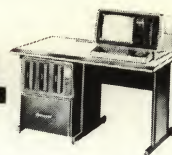
A = 3

Exit Conditions

NZ = Error

A = Error Code

(HL) = Address of deleted **BREAK** key processing program, if HL = 0 on entry



DISKID (function code 15)

This routine reads the diskette ID from any or all of drives 0 through 3. (The diskette ID is assigned by the FORMAT and BACKUP utilities.) This routine is useful when the program needs to ensure that the Operator has inserted the proper diskette.

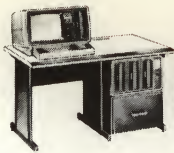
Entry Conditions

- B** = Drive Select Code. If B = 0, read from drive 0, etc.
 B must be one of the following: 0, 1, 2, 3, or 255. If B = 255, then routine reads from all four drives.
- (HL)** = Buffer to hold the diskette ID(s).
 If B = 0, 1, 2 or 3, then buffer must be 8-bytes long. If B = 255, then buffer must be 32 bytes long. Drive 0 ID will be placed in first 8 bytes, then drive 1, etc.
- A** = 15

Exit Conditions

The Diskette ID(s) are placed in the buffers pointed to by register-pair HL. If a drive is not ready, blanks are placed into the buffer.

- NZ** = Error.
- A** = Error Code.



TIMER **(function code 25)**

This routine lets you start a timer to interrupt a program when time runs out. Unlike the DELAY routine, TIMER runs concurrently with your program. One application would be to give an operator a specified number of seconds for keyboard input, and to interrupt the keyboard input routine if no input was made within the time limit.

When setting the timer, you tell it how many seconds to count down. TRSDOS will then continue executing your program, until the timer counts down to zero or you reset the timer.

This is a "one-shot" timer. When it counts to zero and causes an interrupt, it automatically shuts off.

See Programming with TRSDOS for information on interrupts.

Entry Conditions

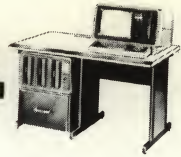
(HL) = Routine to handle interrupt when timer counts to zero

BC = Number of seconds to count down

A = 25

If HL and BC both equal zero, then timer is turned off.

If HL = 0 and BC is not equal to zero, then time count is reset to the value in BC, and timing continues.



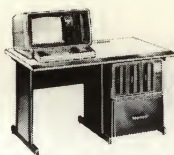
JP2DOS

Jump to DOS (function 36)

This program simply returns control to the command level (TRSDOS READY).
All Open files are Closed automatically.

Entry Conditions

A = 36

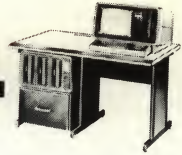


DOSCMD DOS Command (function code 37)

This routine sends TRSDOS a command. After the command is executed, control returns to TRSDOS (TRSDOS READY). All Open files are closed automatically.

Entry Conditions

(HL) = TRSDOS command string
B = Length of command string
A = 37



RETCMD

Return after Command (function code 38)

This routine sends TRSDOS an operator command. After completion of the command, control returns to your program. All Open files are Closed automatically.

Note: Take care that TRSDOS doesn't overlay your program while loading the command file you specified. Most TRSDOS library commands use memory below X'27FF'; a few go up to but not including X'2FFF'. Single-drive, single-disk copies use all user memory. See **Library Commands** for details.

Entry Conditions

(HL) = TRSDOS command string

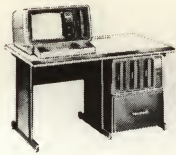
B = Length of command string.

A = 38

Exit Conditions

NZ = Error

A = Error Code



ERROR **(function code 39)**

This routine displays the message ERROR followed by the specified error code. The message appears at the current cursor position.

Entry Conditions

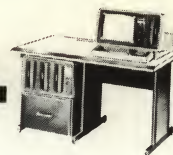
B = Error Code

A = 39

Exit Conditions

NZ = Error

A = Error Code



ERRMSG

Error Message (function code 52)

This routine returns an 80-byte descriptive error message to the specified buffer area. (See list of Error Codes and Messages.)

Entry Conditions

B = Error Code corresponding to message

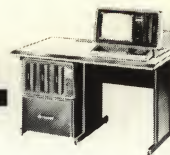
(HL) = 80-byte buffer area in user area (above X'27FF')

A = 52

Exit Conditions

NZ = Error

A = Error Code



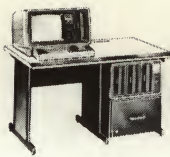
Keyboard

Supervisor calls described in this section*:

FUNCTION CODE	NAME	PURPOSE
1	KBINIT	Clears stored keystrokes.
4	KBCHAR	Gets a character from keyboard.
5	KBLINE	Gets a line from keyboard.
12	VIDKEY	Display message and get line from KB.

Table 4.4. Keyboard Supervisor Calls

*VIDKEY is described later on under "Video Display."



KBINIT

Keyboard Initialize (function code 1)

This routine initializes the keyboard input driver. This call should be made before you start keyboard input. It clears all previous keystrokes.

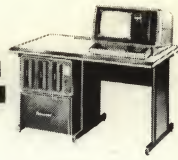
Entry Conditions

A = 1

Exit Conditions

NZ = Error

A = Error Code



KBCCHAR

Keyboard Character (function code 4)

This routine gets one character from the keyboard. The routine returns immediately either with or without a character in register B.

The **BREAK** key is masked from the user — it will never be returned, since it is intercepted by the System. If a SETBRK routine is enabled, control passes to the processing program (see SETBRK) whenever **BREAK** is pressed. Otherwise, control pass to TRSDOS READY.

Entry Conditions

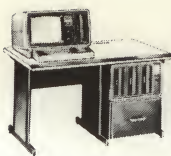
A = 4

Exit Conditions

B = Character found, if any. Only codes within the range [0,127] can be returned. If no character is returned, B is unchanged.

NZ = No character present

A = Error Code



KBLINE Keyboard Line (function code 5)

This routine inputs a line from the Keyboard into a buffer, and echoes the line to the Display, starting at the current cursor position. As each character is received and displayed, the cursor advances to the next position (Scroll Mode – see “Video Display” section on the following pages).

On entry to this routine, the input buffer is filled with periods, and these periods appear on the Display, indicating the length of the input field for the operator’s convenience.

The line ends when a carriage return is typed or when the input buffer is filled. A carriage return and erase-to-end-of-screen are always sent to the Display upon termination of line input; a carriage-return is stored only if the Operator actually pressed **ENTER**.

Entry Conditions

(HL) = Start of input buffer.

B = Maximum number of characters to receive, $0 < B$

A = 5

Exit Conditions

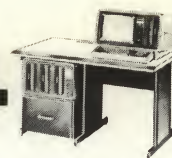
B = Actual number of characters input, including carriage return

C = 0 if input buffer was filled without carriage return. If line ended with a carriage return, then $C = X'0D'$.

Control codes **not** listed below are placed in the buffer and represented on the display with \pm symbols,

KEY	HEX CODE	FUNCTION
BACKSPACE	08	Backspaces the cursor and erases a character.
ENTER	0D	Terminates line. Clears trailing periods on display but not in buffer.
CTRL W	17	Fills remainder of input buffer with blanks, blanks remainder of Display line.
CTRL X	18	Fills remainder of input buffer with blanks, blanks to end of Display.
ESC	1B	Reinitializes input function by filling input buffer with periods and restoring cursor to original position.
←	1C	Backspaces the cursor to allow editing of line. Does not erase characters.
→	1D	Advances the cursor to allow editing of line. Does not erase characters.

Table 4.5. Received Control Codes, code < 32



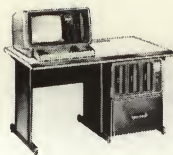
Video Display

Supervisor Calls described in this section:

FUNCTION CODE	NAME	PURPOSE
7	VDINIT	Initializes Display
8	VDCHAR	Sends a character, Scroll Mode
9	VDLINE	Sends a line, Scroll Mode
10	VDGRAF	Sends characters, Graphics Mode
11	VDREAD	Reads characters, Graphics Mode
12	VIDKEY	Displays message, and gets line from KB
26	CURSOR	Turns cursor on or off
27	SCROLL	Sets number of lines at top of display which are not scrolled

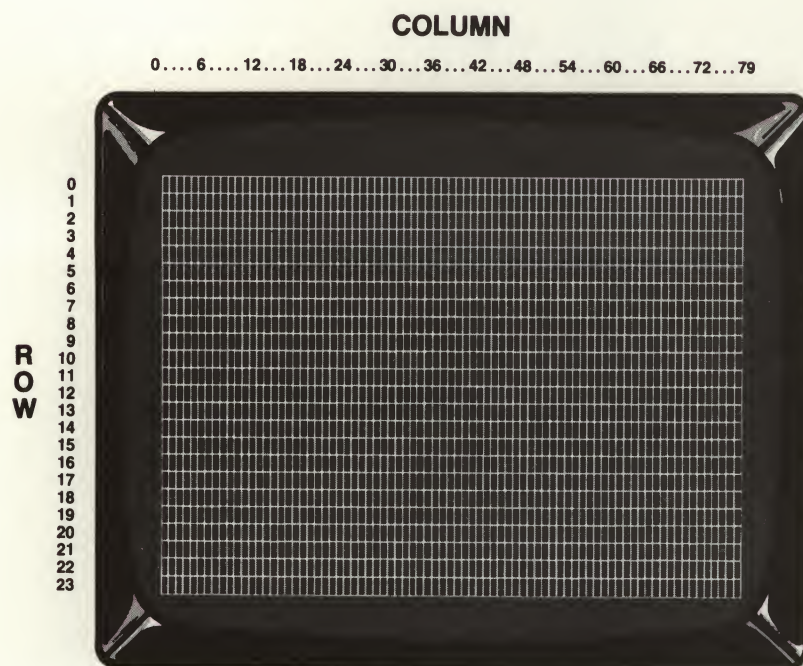
Table 4.6. Video Display Supervisor Calls

The Display has two modes of operation — Scroll and Graphics. Cursor motion and allowable input characters are different in the two modes.



Graphics Mode

In the Graphics Mode, the Display can be thought of as an 80 by 24 matrix, as illustrated below:



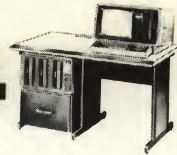
DISPLAY POSITIONS, GRAPHICS MODE

Note: The Display has two character sizes: 80 characters per line and 40 characters per line. The illustration above shows the 80 character per line mode.

Each time an acceptable display character is received, it is displayed at the current cursor position (which is set on entry to the Graphics Mode routines). Before displaying the next character, the cursor position is advanced, as follows:

- If the cursor is to the left of Column 79, it advances to the next column position on the same row.
- If the cursor is at Column 79, it wraps around to Column 0 on the next row.

Cursor motion works the same way in all directions. For example, if the cursor is at Row 23, Column 40, and the X'FF' (graphics-down) code is received, the cursor wraps around to Row 0 in the same column.



Scroll Mode

In the Scroll Mode, the Display can be thought of as a sequence of 1920 display positions, as illustrated below:

Line 0	0, 1, 2, 3, 4, 5, 6	78, 79
Line 1	80, 81, 82, 83	158, 159
Line 2		
Line 3		
Line 4		
Line 5		
Line 6		
Line 7		
Line 8		
Line 9		
Line 10		
Line 11		
Line 12		
Line 13		
Line 14		
Line 15		
Line 16		
Line 17		
Line 18		
Line 19		
Line 20		
Line 21		
Line 22	1760, 1761	1838, 1839
Line 23	1840, 1841	1918, 1919

DISPLAY POSITIONS, SCROLL MODE

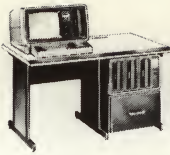
Note: The Display has two character sizes: 80 characters per line and 40 characters per line. The illustration above shows the 80 character per line mode.

In the scroll mode, each time an acceptable display character is received, it is displayed at the current cursor position, and the cursor advances to the next higher numbered position.

When the cursor is on the bottom line and a line-feed or carriage return is received, or when the bottom line is filled, the entire Display is "scrolled":

1. Line 0 is deleted
2. Lines 1-23 are moved up on one line
3. Line 23 is blanked
4. The cursor is set to the beginning of line 23.

Note: Lines 0 to 22 on the Display can be protected from scrolling via the SCROLL function call.



VDINIT

Video Initialization (function code 7)

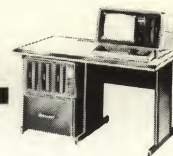
Call this initialization routine once before starting any I/O to the Display. It blanks the screen and resets the cursor to the top left corner (position 0 in the Scroll Mode illustration).

Entry Conditions

- B = Character size switch. If B=0 then size is set to 40 characters/line. Otherwise, size is set to 80 characters/line.
- C = Normal/Reverse switch. If C = 0 then sets Reverse mode, black on white background. Otherwise sets Normal mode, white on black background.
- A = 7

Exit Conditions

- NZ = Error
- A = Error Code



VDCHAR

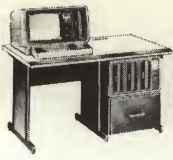
Video Character (function code 8)

This routine outputs a character to the current cursor position. It is a Scroll Mode routine, as described above.

Control Codes not listed below are ignored.

KEY	HEX CODE	FUNCTION
F1	01	Blinking cursor on.
F2	02	Cursor off.
CTRL D	04	Turns on steady cursor.
BACKSPACE	08	Moves cursor back one position and blanks the character at that position.
TAB	09	Advances cursor to next tab position. Tab positions are at 8-character boundaries, 8, 16, 24, 32, ...
CTRL J	0A	Line feed—cursor moves down to next row, same column position.
ENTER	0D	Moves cursor down to beginning of next line.
CTRL W	17	Erases to end of line, cursor doesn't move.
CTRL X	18	Erases to end of screen, cursor doesn't move.
CTRL Y	19	Sets Normal Display mode (white on black). Remains Normal until reset by programmer.
CTRL Z	1A	Sets Reverse Display mode (black on white). Remains Reverse until reset by programmer.
ESC	1B	Erases screen and homes cursor (position 0).
←	1C	Moves cursor back one position.
→	1D	Moves cursor forward one position.
↑	1E	Sets 80 character/line and clears Display.
↓	1F	Sets 40 character/line and clears Display.

Table 4.7. Received Control Codes, Code<X'20'



Model II TRSDOS

Entry Conditions

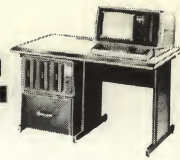
B = ASCII code for character to be output to the Display; character codes **must** be in the range [0,127]

A = 8

Exit Conditions

NZ = Error

A = Error Code



VDLINE Video Line (function code 9)

This routine writes a buffer of data to the Display, starting at the current cursor position. It is a Scroll Mode routine.

The buffer should contain ASCII codes in the range [0,127].

Received Control Codes, code X'20'.

Same as for VDCHAR.

Entry Conditions

- (HL) = Beginning of the buffer containing characters to be set to the Display
- B = Number of characters to be sent
- C = End of line character. This character will be sent to the Display after the buffer text
- A = 9

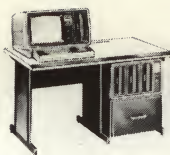
Exit Conditions

- NZ = Error
- A = Error Code

In case of an error:

- B = Number of characters **not** displayed, including the one causing the error
- C = Character causing the error

Upon return, the cursor is always set to the position following the last character displayed.



VDGRAF Video Graphics (function code 10)

This function displays a buffer of characters, starting at a specified row and column. It is a Graphics Mode routine (the cursor “wraps” the Display).

Displayable Characters

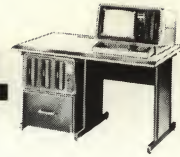
This routine lets you display the 32 graphics characters (and their reverse images). The codes are numbered from 0 through X'1F', and are pictured in the **Operator's Manual**. Codes X'20' through X'7F' are displayed as standard ASCII characters.

In addition, several special control codes are available:

HEX CODE	FUNCTION
F9	Sets Normal (white on black) mode. Cursor does not advance. Mode reverts to previous state after completion of VDGRAF call.
FA	Sets Reverse (black on white) mode. Cursor does not advance. Mode reverts to previous state after completion of VDGRAF call.
FB	Homes cursor (Row 0, Column 0).
FC	Moves cursor back one space. Col.=Col-1. When column equals 79, cursor wraps to Col. 0 on the preceding row.
FD	Moves cursor forward one space. Col.=Col.+1. When column equals 0, cursor wraps to Col. 79 on the next row down.
FE	Moves cursor up one row. Row=Row-1. “Wraps” to Row 23 when Row=0.
FF	Moves cursor down one row. Row=Row+1. “Wraps” up to Row 0 when Row=23.

Table 4.8. Special Graphics Control Codes

At exit, the cursor is always set to the Graphics position immediately after the last character displayed. If the Buffer length was zero, the cursor is set to position specified in BC registers.



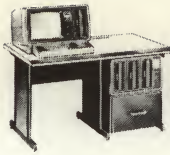
Entry Conditions

- B = Row on screen to start displaying the buffer, $B < 24$. If $B > 23$, then $B \text{ modulo } 24^*$ is used as row position.
- C = Column on screen to start displaying the buffer. In 80 character/line mode, $C < 80$. For $C > 79$, $C \text{ modulo } 80$ is used as column position.
- In 40 character/line mode, $C < 40$. For $C > 39$, $C \text{ modulo } 40$ is used as column position.
- D = Length of buffer, in range $[0, 255]$
- (HL) = Beginning of text buffer. The buffer should contain codes below X'80' or the special control codes above X'F8'. Any value outside these ranges will cause an error.
- A = 10

Exit Conditions

- NZ => Error (invalid character sent)
- A = Error Code

**Modulo*—A cyclical counting system. For modulus n , $x \text{ modulo } n$ is the integer remainder after division of x by n . For example, $85 \text{ modulo } 80 = 5$.



VDREAD Video Read (function code 11)

This routine reads characters from the Video Display into a specified buffer. It is a Graphics Mode routine; when it reads past the last column, it wraps back to column 1 on the next row. When it reads past column 79 on row 23, it wraps back to row 0, column 0.

Reverse (black on white) mode characters are read in as ASCII codes just like their Normal counterparts; Reverse mode is indicated when the most significant bit (bit 7) is set.

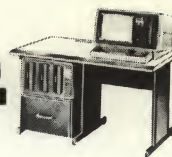
This routine can also be used just to locate the cursor (see below).

Entry Conditions

- B = Row on screen where read starts, $B < 24$
If $B > 23$, then $B \bmod 24$ is used as row position.
- C = Column on screen where read starts
In 80 character/line mode, $C < 80$
For $C > 79$, $C \bmod 80$ is used as column position.
In 40 character/line mode, $C < 40$
For $C > 39$, $C \bmod 40$ is used as column position.
- D = Length of buffer, in range $[0,255]$. If $D = 0$, then B and C are ignored. Current cursor position will be returned as row, column in BC register pair.
- (HL) = Beginning of text buffer
- A = 11

Exit Conditions

- BC = Current cursor position, B = row, C = column. CURSOR position at exit is the same as at entry – VDREAD does not change it.
- NZ = Error
- A = Error Code



VIDKEY (function code 12)

This routine sends a prompting message to the Display and then waits for a line from the Keyboard. It is a Scroll Mode routine, combining the functions of VDLIN and KBLIN.

The routine writes the specified text buffer to the Display, starting at the current cursor position. The text buffer must contain codes <X'80'. Refer to VDLIN for a list of Received Control Codes and other details.

After the Video write, the cursor will be positioned immediately after the last character displayed. (To move it to another position, control codes can be placed at the end of the text buffer.)

Next, the routine gets a line from the Keyboard.

Note: Before starting the line input, all previously stored keystrokes are cleared.

Refer to KBLIN for a list of Received Control Codes and other details.

Entry Conditions

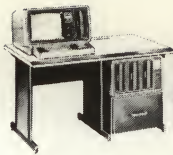
- (HL) = Beginning of text buffer containing display message
- B = Number of characters to be displayed, B in the range [0,255]
- C = Length of Keyboard Input field, C in the range [0,255]
- (DE) = Beginning of text buffer where Keyboard input will be stored
- A = 12

Exit Conditions

- NZ = Error (illegal value in display buffer)
- A = Error Code

If Z is set (no error), then registers B and C contain:

- B = Number of characters input from Keyboard, including carriage return, if any
- C = Keyboard Line termination. If C = 0, then input buffer was filled. Otherwise C = control character that terminated the line (carriage return).

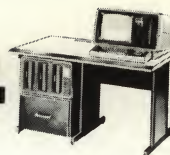


Model II TRSDOS

If Z is not set (error), the registers B and C contain:

B = Number of characters not displayed, including the one causing the error

C = Character causing the error



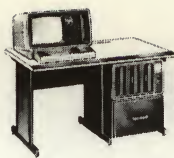
CURSOR **(function code 26)**

This routine turns the blinking cursor on or off. The System still keeps track of the current cursor position, whether it is on or off.

Entry Conditions

B = Function Switch. If B = 0 then cursor will be turned off. If B < > 0 then cursor will be turned on.

A = 26



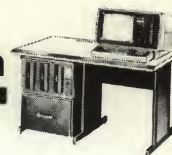
SCROLL **(function code 27)**

This routine lets you protect a portion of the Display from scrolling. From 0 to 22 lines at the **top** of the Display can be protected; when scrolling occurs, only lines below the protected area will be changed.

Entry Conditions

B = Number of lines to be protected, in range [0,22].

A = 27



Line Printer

Supervisor Calls described in this section:

FUNCTION CODE	NAME	PURPOSE
17	PRINIT	Initializes the Line Printer Driver.
18	PRCHAR	Sends a character to the Printer.
19	PRLINE	Sends a line to the Printer.

Table 4.9. Line Printer Supervisor Calls



PRINT

Printer Initialization (function code 17)

This routine initializes the Line Printer driver. It is automatically called when the system is initialized. You don't need to call it unless:

- You want to change some of the parameters.
- The Printer was not available (on-line) when the System was initialized.

When initialized by the System, the following parameters are set:

Page Length (lines to a page)	: 66
Printed Lines per page	: 60
Automatic Form Feed	: Yes
Line Length (Characters/Line)	: 132

Note: Two line feeds are done by the System during initialization. You may want to reset the paper before using the Printer for the first time.

Entry Conditions

B = Page Length (66 is standard)

C = Printed Lines per page (60 is standard). If C = 0, no automatic form feed is done. Otherwise, automatic form feed is done after C lines have been printed.

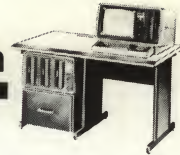
D = Maximum number of characters in a line (132 is standard)

A = 17

Exit Conditions

NZ = Error

A = Error Code



PRCHAR

Print Character (function code 18)

This routine sends one character to the Printer.

Note: Most printers do not print until their buffer is filled or a carriage return is received.

Entry Conditions

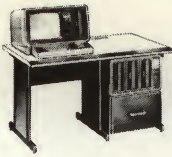
B = ASCII code for character to send

A = 18

Exit Conditions

NZ = Error

A = Error Code



PRLINE **Print Line (function code 19)**

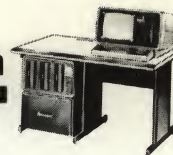
This routine sends a line to the Printer. The line can include control characters as well as printable data. A tab character embedded in the buffer will cause the Printer to skip over to the next 8-character boundary.

Entry Conditions

(HL) = Start of text buffer containing data and controls to send to Printer
B = Length of buffer (number of characters to send)
C = Control Character (any character) to send after last character in buffer
A = 19

Exit Conditions

NZ = Error
A = Error Code

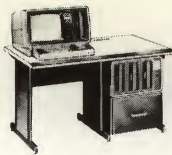


File Access

Supervisor calls described in this section:

Function Code	Name	Function
33	LOCATE	Returns the current record number.
34	READNX	Gets next record (Sequential Access).
35	DIRRD	Reads specified record (Direct Access).
40	OPEN	Sets up access to new or existing file.
41	KILL	Deletes the file from the directory.
42	CLOSE	Terminates access to an Open file.
43	WRITNX	Writes next record (Sequential Access).
44	DIRWR	Writes specified record (Direct Access).

Table. 4.10. File Access Calls



LOCATE (function code 33)

This function returns the number of the current record, i.e., the number of the last record accessed. You can use this call only with Fixed Length Record files.

Entry Conditions

(DE) = Data Control Block for currently Open file (see OPEN)

HL = Reserved for use in later versions of TRSDOS

A = 33

Exit Conditions

BC = Current Record Number

NZ = Error

A = Error Code

Note: In a future release of TRSDOS, (BC) = address of a four-byte value specifying a record number.



READNX

Read Next Record (function code 34)

This routine reads the next record after the current record. (Current record is the last record accessed.) If the file has just been Opened, READNX will read the first record.

Entry Conditions

(DE) = Data Control Block for currently Open file (see OPEN)

HL = Reserved for use in later versions of TRSDOS

A = 34

Exit Conditions

NZ = Error

A = Error Code

Upon return, your record is in the Record Area pointed to by RECADR in the parameter list, or, if RL=256 and record type is Fixed, your record is in the area pointed to by BUFADR.



DIRRD

Direct Read (function code 35)

This routine reads the specified record, allowing direct access.

Note: With VLR files, you can only use it to read the first record or to read the end of file.

Entry Conditions

(DE) = Data Control Block for currently Open file (see OPEN)

BC = Desired record number

BC = 0 means position to beginning of file

BC = X'FFFF' means position to end of file

HL = Reserved for use in later versions of TRSDOS

A = 35

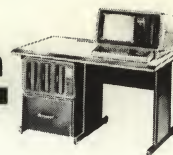
Note: In a future release of TRSDOS, (BC) = address of a four-byte value specifying a record number.

Exit Conditions

NZ = Error

A = Error Code

Upon return, your record will be in the Record Area pointed to by RECADR in the parameter list, or, if RL=256 and record type is Fixed, your record is in the area pointed to by BUFADR.



OPEN (function code 40)

This one call handles both the creation and opening of files.

A given file can only be Open under one Data Control Block at a time.

Because of the versatile file processing routines, this one DCB is sufficient to handle the various I/O applications.

Entry Conditions for OPEN

(DE) = 60-byte Data Control Block (see below)

(HL) = 11-byte Parameter List (see below)

A = 40

Exit Conditions

NZ = Error

A = Error Code

Before calling OPEN, you must reserve space for the Data Control Block, Parameter List, Buffer Area and Record Area, as described below.

Data Control Block (60 bytes)

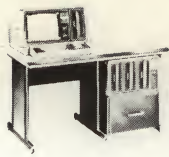
The Data Control Block (DCB) is used by the System for file access bookkeeping. You will also use it to pass the filespec for the file you want to Open, as follows:

Before calling OPEN, place the filespec at the beginning of the DCB, followed by a carriage return. See **File Specification** in Section 1 of this manual.

For example (\$ signifies a carriage return):

CONTENTS OF FIRST BYTES OF DCB BEFORE OPEN

FILENAME / EXT . PASSWORD : d (DISKETTE) \$



Model II TRSDOS

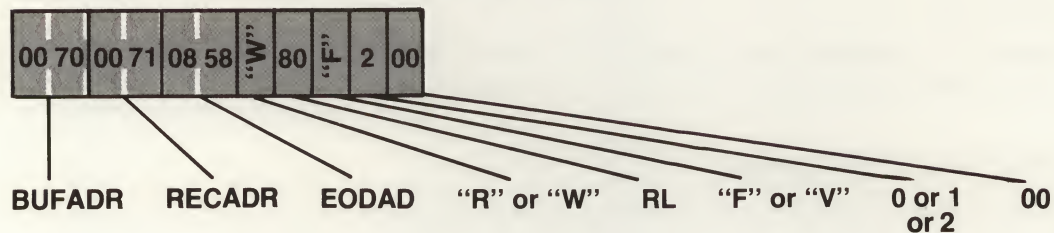
While a file is Open, the filespec is replaced with information used by the System for bookkeeping. When the file is Closed, the original filespec (except for the password) will be put back into the DCB.

Important Note: Do not ever modify any portion of the DCB while the file is Open. If you do, the results will be unpredictable.

Parameter List (11 bytes)

This list contains information TRSDOS needs to create or access the file:

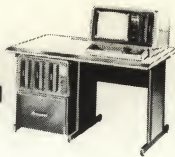
CONTENTS OF PARAMETER LIST (SAMPLE)



BUFADR (Buffer Address). This two-byte field must point to the beginning of the Buffer Area.

The Buffer Area is the space TRSDOS will use to process all file accesses. If spanned records are possible, you must reserve 512 bytes. If no spanning is possible, reserve only 256 bytes.

With Fixed Length Record files, spanning is only required when the record length is not an even divisor of 256. For example, if the record length is 64, then each physical record contains four records exactly, and no spanning is required. In this case, reserve only 256 bytes for processing.



However, if the record length is 24 (not an even divisor of 256), then some records will have to be spanned. In this case, you will need to reserve 512 bytes.

With Variable Length Record files, you must always reserve 512 bytes for processing. This is because spanning may be required, depending on the lengths of the individual records in the file.

RECADR (Record Address). This two-byte field must point to the beginning of the Record Area.

For disk reads, this is where TRSDOS will place the record. For disk writes, this is where you put the record to be written.

Exception: For FLR files with a record length of 256, this address is not used. Your record will be in the buffer area pointed to by BUFADR.

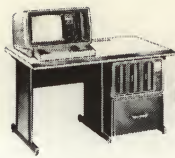
For Fixed Length Record files with record length not equal to 256, this buffer should be the same size as the record length. For Variable Length Files, this area should be long enough to contain the longest record in the file (including the length-byte). If you are not sure what the longest record will be, reserve 256 bytes.

EODAD (End of Data Address). This two-byte field can be used to give TRSDOS a transfer address to use in case the end of file is reached during an attempted read; control will transfer to the EODAD address if the end of file is reached during a read operation. If EODAD = 0 and end of file is reached, the SVC will simply return with the end of file error code in register A.

“R” or “W” (Read or Write). Put an ASCII “R” here to allow read-access only; put an ASCII “W” here to allow read and/or write access.

RL (Record Length). This one-byte field specifies the record length to be used. Zero indicates a record length of 256. For Variable Length record files, this field is ignored. If the file already exists, and the Creation Code is 0, the System will supply the correct RL value, regardless of what you put there.

“V” or “F” (Variable or Fixed Length). This one-byte field contains either an ASCII “V” for Variable or an ASCII “F” for Fixed. Once a file has been created, this attribute cannot be changed. If the file already exists, and the Creation Code is 0, the System will supply the correct “F” or “V” value, regardless of what you put in the parameter list.



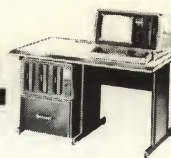
Model II TRSDOS

0 or 1 or 2 (Creation Code). This one-byte field contains a binary number 0, 1 or 2.

CODE	MEANING
0	Open the file only if it already exists. Do not create a new file in directory. Record Length and end of file are not reset.
1	Create a new file only; do not Open an existing file. Record Length and end of file are set at Open time.
2	Open existing file; if file not found, create it (record length and end of file will be reset).

Table 4.11. File Creation Code

00 (End-of-List Marker). Always put a binary zero at the end of the Parameter List.



KILL **(function code 41)**

This routine deletes the specified file from the directory. A file must be Closed before it can be Killed.

Entry Conditions

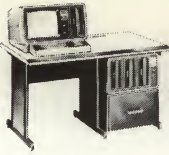
(DE) =Data Control Block, containing standard TRSDOS filespec (see illustration in description of OPEN)

A = 41

Exit Conditions

NZ = Error

A = Error Code



CLOSE **(function code 42)**

This routine terminates access to the file. If there are records in the Buffer Area not yet written, they will be written at this time.

Entry Conditions

(DE) = Data Control Block for currently Open file

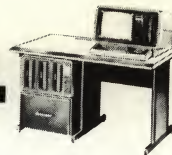
A = 42

Exit Conditions

NZ => Error

A = Error Code

Upon return, the filespec (except for the password) will be put back into the DCB.



WRITNX

Write Next Record (function code 43)

This routine writes the next record after the last record accessed; that is, it writes sequentially. If WRITNX is the first access after the file is Opened, the first record will be written.

Entry Conditions

- (DE) = Data Control Block for currently Open file
- (HL) = Reserved for use in later versions of TRSDOS
- A = 43

Before calling WRITNX, put your record in the record area pointed to by RECADR in the Parameter List, or, if RL=256 and record type is Fixed, your record is in the area pointed to by BUFADR.

Exit Conditions

- NZ = Error
- A = Error Code



DIRWR

Direct Write (function code 44)

This routine writes the specified record. It writes your record into the specified record position of the file.

Note: For VLR files, you can only position to the beginning or end of file. When you write to a VLR file, the end of file is reset to the last record written.

Entry Conditions

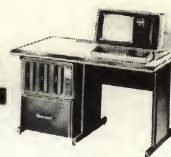
(DE) = Data Control Block for currently Open file
BC = Record number you want to write
 BC = 0 means write first record in file
 BC = X'FFFF' means write record at end of file
(HL) = Reserved for use in later versions of TRSDOS
A = 44

Before calling DIRWR, put your record into the Record Area pointed to by RECADR in the Parameter list, or, if RL=256 and record type is Fixed, put record in area pointed to by BUFADR.

Exit Conditions

NZ = Error
A = Error Code

Upon return, the current record pointer is incremented by one.

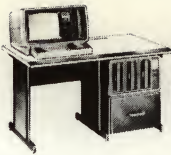


Computational

Supervisor calls described in this section:

Function Code	Name	Function
6	DELAY	Provides a delay-loop
20	RANDOM	Provides a random number, range [0,254]
21	BINDEC	Converts binary to ASCII-coded decimal, and vice versa
22	STCMP	Compares two text strings
23	MPYDIV	Performs 8 bit * 16 bit multiplication and 16 bit / 8 bit division
24	BINHEX	Converts binary to ASCII-coded hexadecimal, and vice-versa
45	DATE	Sets or returns the time and date.
48	PARSER	Finds the alphanumeric parameter field in a text string
49	STSCAN	Looks for a specified string inside a text buffer

Table 4.12 Computational Supervisor Calls



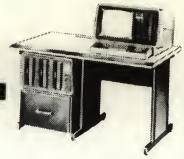
DELAY (function code 6)

This routine provides a delay routine, returning control to the calling program after the specified time has elapsed.

Entry Conditions

BC = Delay Multiplier. If $BC = 0$, then delay time will be 426 milliseconds. If $BC > 0$, then delay time will be: $6.5 * (BC - 1) + 22$ microseconds.

A = 6



RANDOM

(function code 20)

This routine returns a random one-byte value. To extend the cycle of repetition, the instantaneous time/date are used in generating the number.

You pass the routine a limit value; the value returned is in the range $[0, \text{limit} - 1]$. For example, if the limit is 255, then the value returned will be in the range $[0, 254]$.

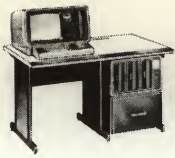
Entry Conditions

B = Limit value

A = 20

Exit Conditions

C = Random number
For $B > 1$, number returned is in range $[0, B - 1]$
For $B = 0$ or 1, number returned = 0



BINDEC **Binary/Decimal (function code 21)**

This routine converts a two-byte binary number to ASCII-coded decimal, and vice versa. Decimal range is [0,65535].

Entry Conditions

B = Function Switch:
If **B** = 0, then convert binary to ASCII decimal
If **B** is not 0, then convert ASCII decimal to binary

Contents of other registers when **B** = 0, binary to decimal:

DE = Two-byte binary number to convert

(**HL**) = 5-byte area to contain ASCII-coded decimal value upon return. The field will contain decimal digits (X'30'–X'39'), leading zeroes on the left as necessary to fill the field; for example, the number 21 would be:

00021

Contents of other registers when **B** is not 0, decimal to binary:

(**HL**) = 5-byte area containing ASCII decimal value to be converted to binary

A = 21

Exit Conditions

(**HL**) = Decimal value

DE = Binary value



STCMP

String Comparison (function code 22)

This routine compares two strings to determine their collating sequence.

Entry Conditions

(DE) = First string
(HL) = Second string
BC = Number of characters to compare
A = 22

Exit Conditions

Status bits indicate results, as follows:

Z flag set indicates strings are identical.

NZ indicates strings not identical.

Carry flag set indicates first string (pointed to by DE) precedes second string (pointed to by HL) in collating sequence.

Other register contents:

A = First non-matching character in first string

When strings are not equal, you can get further information from the prime registers, as follows:

(HL') = Address of first non-matching character in second string

(DE') = Address of first non-matching character in first string

BC' = Number of characters remaining, including the non-matching character



MPYDIV **Multiply Divide (function code 23)**

This routine does multiplication and division with one 2-byte value and one 1-byte value.

Entry Conditions

B = Function Switch:
 If B = 0 then multiply
 If B not 0 then divide
A = 23

For multiplication:

HL = Multiplicand
C = Multiplier

For division:

HL = Dividend
C = Divisor

Exit Conditions

HL = Result (product HL * C or quotient HL/C)
A = Overflow byte (multiplication only)
C = Remainder (division only)

Status bits affected by division:

Carry flag set if dividing by zero. Divide not attempted.
Z flag set only if the quotient is zero.

Status bits affected by multiplication:

Carry Flag set if overflow.
Z flag set only if result is zero.



BINHEX

Binary/Hexadecimal (function code 24)

This routine converts a two-byte binary number to ASCII-coded hexadecimal, and vice versa. Hexadecimal range is [0,FFFF].

Entry Conditions

B = Function Switch:
 If B = 0, then convert binary to ASCII hexadecimal
 If B is not 0, then convert ASCII hexadecimal to binary

A = 24

Contents of other registers when B = 0:

DE = Two-byte binary number to convert

(HL) = 4 byte area to contain ASCII coded hexadecimal value upon return. The field will contain hexadecimal digits with leading zeroes on the left as necessary to fill the field, for example, the number X'FF' would be:
 00FF

Contents of other registers when B not 0:

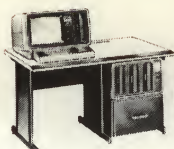
(HL) = 5-byte area containing ASCII hexadecimal value to be converted as described above

A = 24

Exit Conditions

(HL) = Hexadecimal value

DE = Binary value



DATE (function code 45)

This routine sets or returns the real-time (time and date). The data is returned as a 26-byte ASCII string containing 8 fields.

CONTENTS OF TIME/DATE STRING (SAMPLE)

S	A	T	A	P	R	2	8	1	9	7	9	1	1	8	1	3	.	2	0	.	4	2	0	4	5
NAME OF DAY			MON.	DAY OF MON.		YR.	DAY OF YEAR		TIME						MON. #	DAY OF WEEK									

Example Time/Date string:

SATAPR28197911813.20.42045

Represents the data "Saturday, April 28, 1979, 118th day of the year, 13:20:42 hours, 4th month of the year, 5th day of the week.

Notes: DAY OF WEEK Field: Monday is day 0. The date calculations are based on the Julian Calendar.

Entry Conditions

B = Function Switch

If B = 0 (Get time/date):

(HL) = 26-byte buffer where date/time will be stored

If B = 1 (Set date):

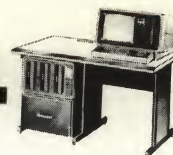
(HL) = 10-byte buffer containing date in this form:

MM/DD/YYYY

If B = 2 (Set time):

(HL) = 8+byte buffer containing time in this form:

HH.MM.SS



PARSER (function code 48)

This general-purpose routine "parses" (analyzes) a text buffer into fields and subfields. PARSER is useful for analyzing TRSDOS command lines including keyword commands, file specifications, keyword options and parameters. It can also be used as a fundamental routine for a compiler or text editor.

By necessity, the description of PARSER is rather long and detailed. In actual use, the routine is as convenient as it is powerful. For example, PARSER is designed to allow repetitive calls for processing a text buffer; on exit from the routine, parameters for the next call are all readily available in appropriate registers.

The routine has pre-defined sets of field-characters and separators; you can use these or re-define them to suit your application.

In general, a field is any string of alphanumeric characters (A-Z, a-z, 0-9) with no embedded blanks. Fields are delimited by separators and terminators, defined below. For example, the line:

BAUD=300, PARITY=EVEN WORD=7

contains 6 fields: BAUD, 300, PARITY, EVEN, WORD, and 7.

However, a field can also be delimited by paired quote marks:
"field" or 'field'

When the quote marks are used, **any** characters, not just alphanumerics, are taken as part of the field. The quote marks are not included in the field. For example, the line:

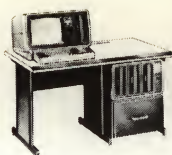
'DATE(07/11/79)'

will be interpreted as one field containing everything inside the quotes. When a quote mark is used to mark the start of a field, the same type of quote mark must be used to mark the end of the field. This allows you to include quotes in a field, for example:

"X'FF00' "

will be parsed as one field containing **everything** inside the double quotes " " , including the single quote marks ' ' .

A separator is any non-alphanumeric character. PARSE will always stop when a separator is encountered, **except** when the separator is a blank (X'20'). Leading and trailing blanks are ignored. After trailing blanks, PARSE stops at the beginning of the next field, or on the first non-blank separator.



Model II TRSDOS

You can also define terminators, which will stop the parse regardless of whether a field has been found. Unless you specifically define these, PARSE will only stop on non-blank separators.

Separators and terminators have the same effect on a parse; the only difference is in how they affect the F (Flag) register on exit.

To re-define the field, separator, and terminator sets

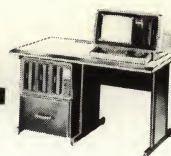
If you need to change the field and separator sets, or define terminators, you can provide three change-lists via a List Address Block, explained later.

Entry Conditions

- (HL) = Text buffer
- (DE) = List Address Block
 - DE = 0 indicates no lists are to be used
- C = Maximum length of parse
- A = 48

Exit Conditions

- (HL) = Field-position:
 - (HL) = First byte of field, if a delimited field was found
 - (HL) = Terminator or non-blank separator if no field was found
 - (HL) = Last byte of buffer if parse reached maximum length
- B = Actual lengths of field, excluding leading and trailing blanks
- A = Character preceding the field just found. If B = 0, A = X'FF'
- C = Number of bytes remaining to parse after terminator or separator.
Note that trailing blanks have been parsed.
- D = Separator or terminator at end of field. If D = X'FF' then parse stopped without finding a non-blank separator or terminator.
- E = Displacement pointer for next parse call. Add E to HL to get:
 - a) Beginning address of next field, or
 - b) Address of byte following the last byte parsed. Note that if parse reached maximum length, then $E + HL = \text{Address following end of text buffer}$. If parse did not reach maximum length, and E = 1, then $E + HL = \text{Address following separator or terminator}$.



Status bits (F register) affected when parse did **not** reach maximum length:

Z flag:

Z (set) if parse ended with a separator

NZ (not set) if parse ended with a terminator

C flag:

C (set) if there were trailing blanks between end of field and next non-blank separator or terminator

NC (not set) if there were no trailing blanks

List Address Block

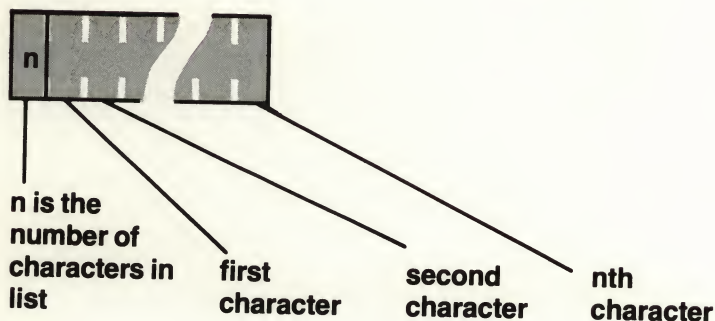
The List Address Block is six bytes long and contains two-byte addresses (lsb,msb) for three change-lists:

List 1: Characters to be used as terminators

List 2: Additions to the set of field characters

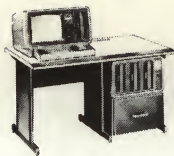
List 3: Deletions from the set of field characters, i.e., alphanumerics to be interpreted as separators

Each list has the following form:



Notes:

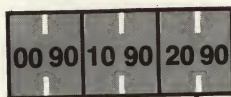
1. There are three ways to indicate a null list:
 - a) Set the character-count byte (n) equal to zero.
 - b) Set the pointer in the List Address Block to zero.
 - c) Set DE=0 if you aren't going to provide any lists.
2. Characters are stored in lists in ASCII form.
3. If a character appears in more than one list, it will have the characteristics of the first list that contains it.



Model II TRSDOS

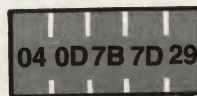
Here is a typical List Address Block with its associated lists. Assume that on entry to PARSE, DE = X'8000'.

X'8000'



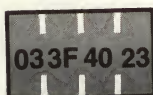
Start of list 1 Start of list 2 Start of list 3

X'9000



4 characters in list 1 carriage return " { " } ") "

X'9010'

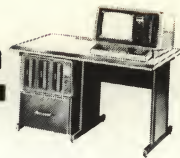


3 characters in list 2 " ? " @ " # "

X'9020



null list



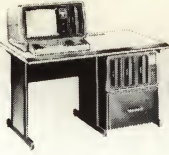
Sample Programming

The following code shows typical repetitive uses of PARSER to break up a parameter list.

```

;-----PREPARE FOR PARSING LOOP-----
      LD      C,MAXLEN      ; C = Maximum length to parse
      LD      E, 0          ; For initial call to NXTFLD
      LD      HL,BUFFER     ; (HL) = string to parse
;-----PARSE LOOP-----
PARSE  CALL    NXTFLD        ; Routine to call PARSER
      CALL    HANDLR        ; Routine to handle new field
      JR      NZ,NXTRTN     ; Go to next routine if
                               ; parsed ended on terminator
      LD      A,C           ; Else get new max length
      OR      A             ; Is it zero?
      JR      NZ,PARSE      ; If not, then continue
      LD      A,0FFH        ;
      CP      D             ; If D=0FFH then no separator
                               ; at end of buffer.
      JR      Z,ERR         ; So go to error routine
      JR      NXTRTN        ; Else, then do next routine.
;-----FIELD-HANDLING ROUTINE-----
HANDLR PUSH    AF           ; Must save status registers
                               ; and any other registers
                               ; will be changed.
                               ;
; Processing code goes here...
      POP     AF            ; Restore AF (and other
                               ; registers saved at entry)
      RET
;-----CALL TO PARSER-----
NXTFLD LD      D,0          ; Zero msb of DE
      ADD     HL,DE         ; (HL) = where to start parse
      LD      DE,LAB        ; (DE) = List address block
                               ; If DE=0 then no lists used.
      LD      A,46          ; Function Code
      RST     8
      RET
;-----PROGRAM CONTINUES HERE-----
NXTRTN EQU     $

```



STSCAN String Scan (function code 49)

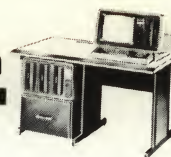
This is a general purpose string scan. It searches through a specified text buffer for the specified string. This string can consist of any values 0-255 (it is not strictly alphanumeric).

Entry Conditions

- (HL) = Text area to be searched
- (DE) = Compare string
- B = Length of compare string
- A = 49

Exit Conditions

- NZ = String not found
- Z = String found
- (HL) = Start position of matching string in search string

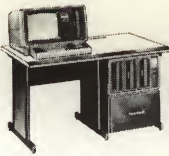


Serial Communications

Supervisor Calls described in this section:

Function Code	Name	Function
55	RS232C	Set or turn off channel A or B for serial input/output.
96	ARCV	Channel A receive
97	ATX	Channel A transmit
98	BRCV	Channel B receive
99	BTX	Channel B receive

These routines allow you to use the Model II's RS-232C interface, channels A and B on the back panel. See the **Model II Operation Manual** for a description of signals available.



RS232C Initialize RS-232C Channel (Function Code 55)

This routine sets up or disables either channel A or B. Before using it, the appropriate channel should be connected to the modem or other equipment.

This routine sets the standard RS-232C parameters, and defines a pair of supervisor calls for I/O to the specified channel. When you initialize Channel A, SVC's 96 and 97 are defined; when you initialize Channel B, SVC's 98 and 99 are defined. See ARCV, ARTX, BRCV, and BTX.

Before re-initializing a channel, **always** turn it off.

Entry Conditions

- (HL) = Parameter list described below
B = Function switch:
 If B is not equal to zero then turn on the channel and define I/O
 SVC's for that channel
 If B is equal to zero then turn off the channel and delete I/O
 SVC's for that channel. In this case only the first byte in the
 parameter list ("A" or "B") is used.
A = 55

Parameter List

This six-byte list includes the necessary RS-232C parameters:

CHANNEL	BAUD RATE	WORD LENGTH	PARITY	STOP BITS	END LIST MARKER
---------	-----------	-------------	--------	-----------	-----------------

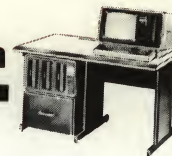
CHANNEL is an ASCII "A" for channel A, or "B" for channel B.

BAUD RATE is a binary value from 1 to 7:

- 1 for 110 baud
- 2 for 150 baud
- 3 for 300 baud
- 4 for 600 baud
- 5 for 1200 baud
- 6 for 2400 baud
- 7 for 4800 baud
- 8 for 9600 baud

WORD LENGTH is a binary value from 5 to 8:

- 5 for 5-bit words
- 6 for 6-bit words
- 7 for 7-bit words
- 8 for 8-bit words



PARITY is an ASCII "E" for even, "O" for odd, or "N" for none.

STOP BITS is a binary 1 for 1 stop bit, or 2 for 2 stop bits.

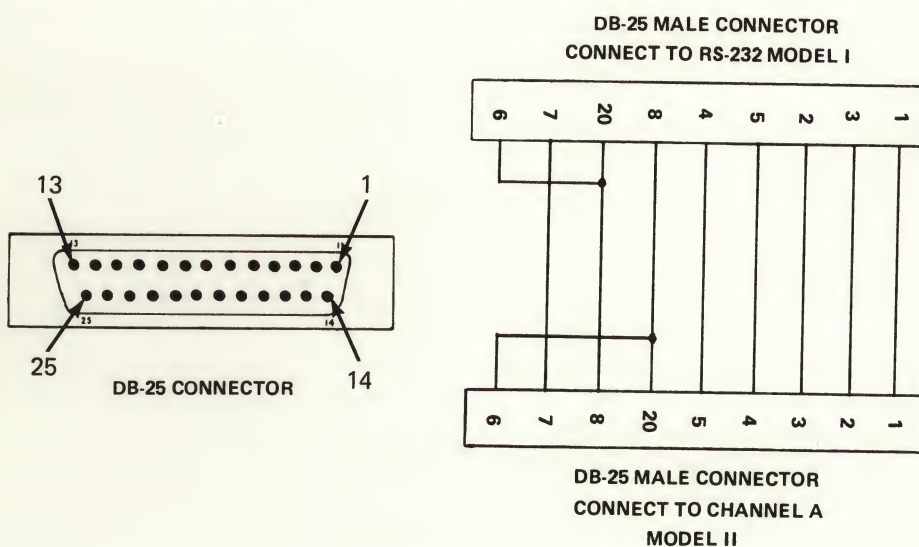
END LIST MARKER is a binary 0.

Exit Conditions

NZ = Error

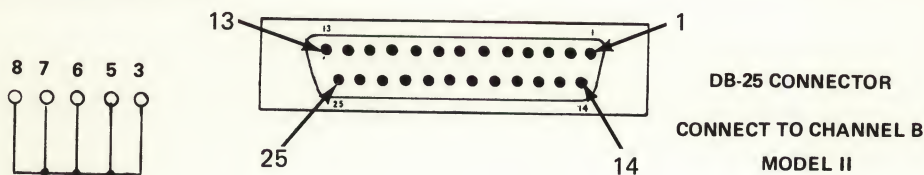
A = Error return code

For hard-wiring between two Model II's without a modem, use the wiring arrangement described below (Model II to Model II **only**)



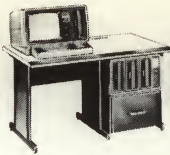
Connection Diagram, Model II (Channel A or B) to Model II (Channel A or B). Use stranded wire, 24-gauge, to connect two DB-25 connectors as illustrated. If wire length exceeds 50 feet, twist lines 7 (GND), 2 (TD) and 3 (RD). Refer to the **Model II Operation Manual** for a description of signals available.

In addition, if you are only going to use one of the serial channels, pins 3, 5, 6, 7, and 8 on the other channel must be tied together **before** you initialize the channels with SETCOM or RS232C. Prepare a DB-25 male terminator plug for the unused channel:



JUMPER DIAGRAM

DUMMY TERMINATOR FOR MODEL II
(SERIAL TERMINATOR PLUG)



ARCV Channel A Receive (function Code 96)

This routine inputs a character from the serial Channel A. It is analogous to keyboard character input (see KBCHAR).

TRSDOS sets up ARCV and ATX when you initialize channel A (see RS232C). If you call this routine without previously initializing channel A, you will get an error return code of 1 (no function exists).

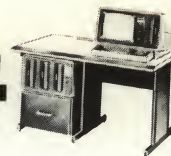
Entry Conditions

A = 96

Exit Conditions

B = Character found, if any
NZ = No character found
C = (Carry Flag Set) Modem carrier not present when SVC was entered
A = Communications status:

BIT	MEANING WHEN SET
0	Not used
1	Not used
2	Not used
3	Modem carrier was lost
4	Parity error occurred on character found in register B.
5	Data lost—more than one character received between SVC's. B. contains last character received.
6	Framing error occurred on last character received.
7	A "break sequence" (extended null character) was received.



ATX Channel A Transmit (function Code 97)

This routine sends a character to the serial Channel A. It is analogous to video character output (see VDCHAR).

TRSDOS sets up ARCV and ATX when you initialize channel A (see RS232C). If you call this routine without previously initializing channel A, you will get an error return code of 1 (no function exists).

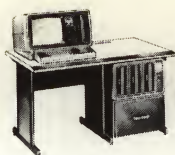
Entry Conditions

B = ASCII code for character to be sent
A = 97

Exit Conditions

NZ = No character sent
C = (Carry Flag Set) Modem carrier not present when SVC was entered
A = Communications status:

BIT	MEANING WHEN SET
0	Clear to Send (CTS) was not detected.
1	Not used.
2	Transmitter is busy.
3	Modem carrier was lost.
4	Not used
5	Not used
6	Not used
7	Not used



BRCV

Channel B Receive (function Code 98)

This routine inputs a character from the serial Channel B. It is analogous to keyboard character input (see KBCHAR).

TRSDOS sets up BRCV and BTX when you initialize channel B (see RS232C). If you call this routine without previously initializing channel B, you will get an error return code of 1 (no function exists).

Entry Conditions

A = 98

Exit Conditions

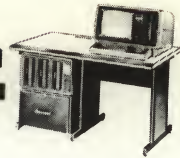
B = Character found, if any.

NZ = No character found.

C = (Carry Flag Set) Modem carrier not present when SVC was entered

A = Communications status:

BIT	MEANING WHEN SET
0	Not used
1	Not used
2	Not used
3	Modem carrier was lost.
4	Parity error occurred on character found in register B.
5	Data lost—more than one character received between SVC's. B contains last character received.
6	Framing error occurred on last character received.
7	A "break sequence" (extended null character) was received.



BTX

Channel B Transmit (function Code 99)

This routine sends a character to the serial Channel B. It is analogous to video character output (see VDCHAR).

TRSDOS sets up BRCV and BTX when you initialize channel B (see RS232C). If you call this routine without previously initializing channel B, you will get an error return code of 1 (no function exists).

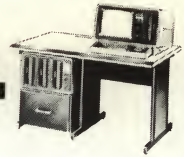
Entry Conditions

B = ASCII code for character to be sent
A = 99

Exit Conditions

NZ = No character sent
C = (Carry Flag Set) Modem carrier not present when SVC was entered
A = Communications status:

BIT	MEANING WHEN SET
0	Clear to Send (CTS) was not detected.
1	Not used
2	Transmitter is busy.
3	Modem carrier was lost.
4	Not used
5	Not used
6	Not used
7	Not used



Programming with TRSDOS

This section tells you how to execute your own machine-language programs under TRSDOS. It includes two sections:

- Program Entry Conditions—how control is transferred to your program after it is loaded from disk.
- Handling Programmed Interrupts—how to write an interrupt service routine for **BREAK** key processing and TIMER interrupts.

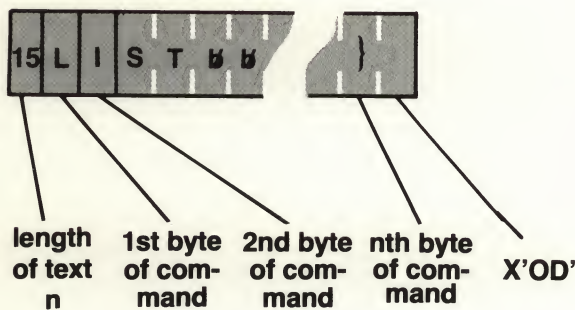
To create and use a program:

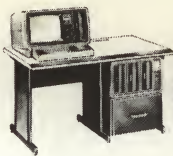
1. Enter the program into memory, either with DEBUG, an assembler, or via the serial interface channel from another device.
2. Use the DUMP command to save the program as an executable disk file, setting load and transfer addresses.
3. To run the program, input the file name to the TRSDOS command interpreter (TRSDOS READY mode).

Program Entry Conditions

Upon entry to your program, TRSDOS sets up the following registers:

- BC** = Address of first byte following your program, i.e., the first free byte for use by your program
- DE** = Highest memory address not protected by TRSDOS, i.e., the end of memory which can be used by your program
- HL** = Address of buffer containing the last command entered to the TRSDOS command interpreter. The first byte of the buffer contains the length of the command line, not including the carriage return. The text of the command follows this length byte. For example:





Handling Programmed Interrupts

TRSDOS allows two user-programmed interrupts as described under SETBRK and TIMER. When either kind of interrupt is received (**BREAK** key is pressed or TIMER counts to zero), control transfer to your interrupt handling routine.

Note: System routines called by your program are also subject to interrupts. Interrupt handlers can also be interrupted.

Upon entry to your interrupt processing routine, TRSDOS sets up the registers as follows:

(SP) = The address of the next instruction to be executed when the interrupt was received

Other registers:

Contents are the same as they were when the interrupt was received.

Before doing any processing, you should save all registers. When finished processing, restore all registers and execute a return to continue with the interrupted program.

It is good practice to keep interrupt handling routines short; ideally, the routine simply flags the main program that an interrupt has occurred and returns. The main program can then respond to the interrupt flag when convenient.

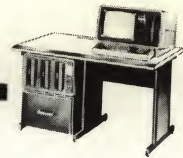
Always end your interrupt handler with the RET instruction and with all registers intact.

TRSDOS is serially reusable but not always re-entrant. More specifically, your interrupt routine should not call TRSDOS SVC's, since under some conditions this will produce unpredictable results.

Section 5

Appendix

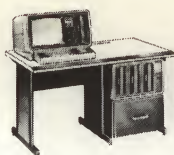
TRSDOS Character Codes
Keyboard Code Map
Decimal-Hexadecimal Conversion



TRSDOS Character Codes

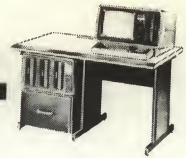
Code		Character		
Dec.	Hex.	Key-board	Video Display	
			Scroll mode	Graphics mode
00	00	HOLD		
01	01	F1	Turns on blinking cursor	
02	02	F2	Turns off cursor	
03	03	BREAK * CTRL C		
04	04	CTRL D	Turns on steady cursor	
05	05	CTRL E		
06	06	CTRL F		
07	07	CTRL G		
08	08	BACKSPACE CTRL H	Backspaces cursor and erases character	
09	09	TAB CTRL I	Advance cursor to next 8-character boundary	
10	0A	CTRL J	Line feed	
11	0B	CTRL K		
12	0C	CTRL L		
13	0D	ENTER CTRL M	Carriage return	
14	0E	CTRL N		
15	0F	CTRL O		
16	10	CTRL P		
17	11	CTRL Q		
18	12	CTRL R		
19	13	CTRL S		
20	14	CTRL T		
21	15	CTRL U		
22	16	CTRL V		
23	17	CTRL W	Erase to end of line	
24	18	CTRL X	Erase to end of screen	
25	19	CTRL Y	Sets white-on-black mode	
26	1A	CTRL Z	Sets black-on-white mode	
27	1B	ESC	Clears screen, homes cursor	
28	1C	←	Moves cursor back	
29	1D	→	Moves cursor forward	
30	1E	↑	Sets 80-character mode and clears display	
31	1F	↓	Sets 40-character mode and clears display	

* **BREAK** is always intercepted. It will never return a 'X'03'.

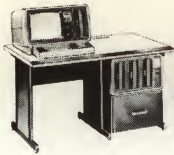


Model II TRSDOS

Code		Character		
Dec.	Hex.	Key-board	Video Display	
			Scroll mode	Graphics mode
32	20	Space Bar	␣	␣
33	21	!	!	!
34	22	"	"	"
35	23	#	#	#
36	24	\$	\$	\$
37	25	%	%	%
38	26	&	&	&
39	27	'	'	'
40	28	(((
41	29)))
42	2A	*	*	*
43	2B	+	+	+
44	2C	,	,	,
45	2D	-	-	-
46	2E	.	.	.
47	2F	/	/	/
48	30	0	0	0
49	31	1	1	1
50	32	2	2	2
51	33	3	3	3
52	34	4	4	4
53	35	5	5	5
54	36	6	6	6
55	37	7	7	7
56	38	8	8	8
57	39	9	9	9
58	3A	:	:	:
59	3B	;	;	;
60	3C	<	<	<
61	3D	=	=	=
62	3E	>	>	>
63	3F	?	?	?
64	40	@	@	@
65	41	A	A	A
66	42	B	B	B
67	43	C	C	C
68	44	D	D	D



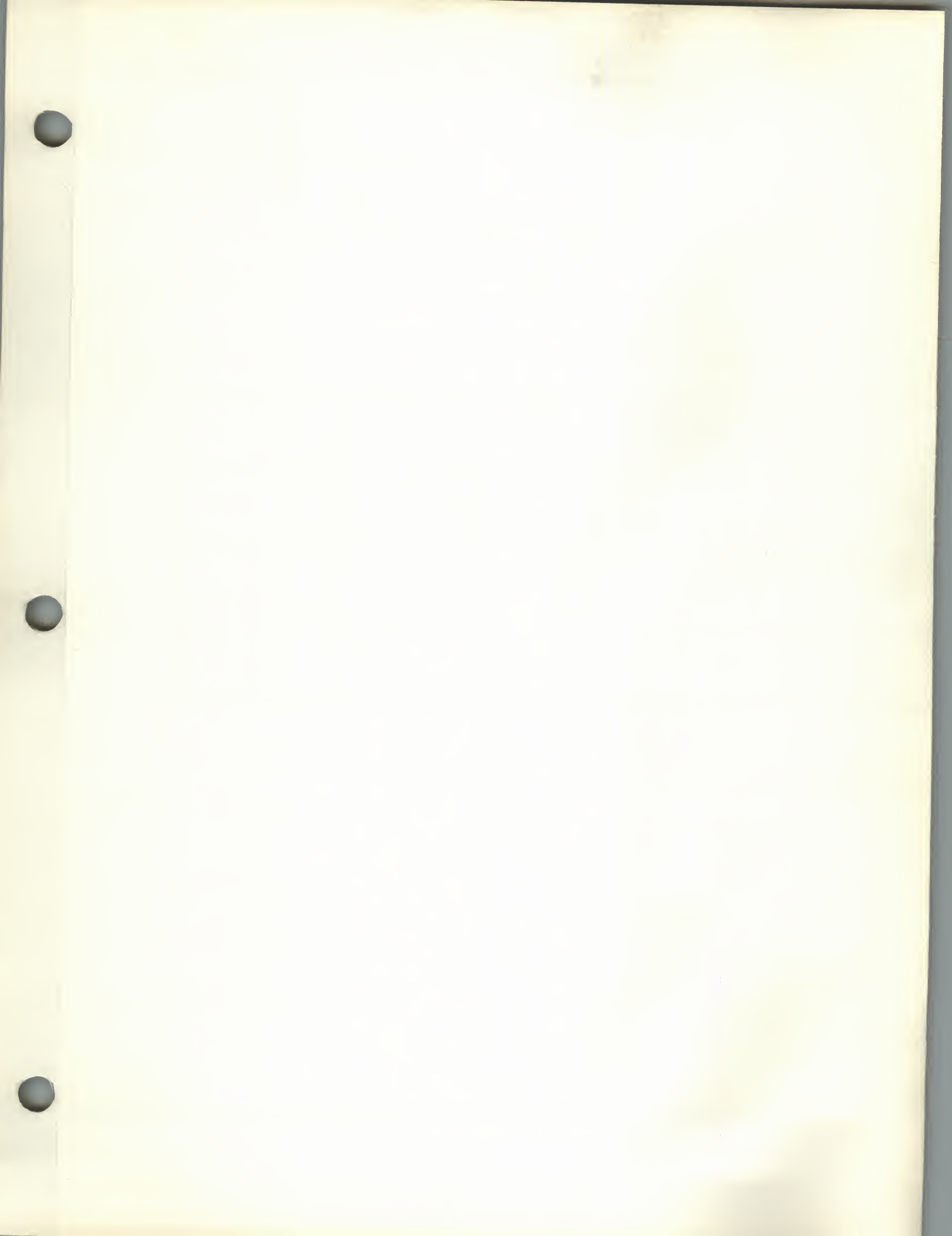
Code		Character		
Dec.	Hex.	Key-board	Video Display	
			Scroll mode	Graphics mode
69	45	E	E	E
70	46	F	F	F
71	47	G	G	G
72	48	H	H	H
73	49	I	I	I
74	4A	J	J	J
75	4B	K	K	K
76	4C	L	L	L
77	4D	M	M	M
78	4E	N	N	N
79	4F	O	O	O
80	50	P	P	P
81	51	Q	Q	Q
82	52	R	R	R
83	53	S	S	S
84	54	T	T	T
85	55	U	U	U
86	56	V	V	V
87	57	W	W	W
88	58	X	X	X
89	59	Y	Y	Y
90	5A	Z	Z	Z
91	5B	□	□	□
92	5C	CTRL 9	\	\
93	5D	□	□	□
94	5E	^	^	^
95	5F	—	—	—
96	60			,
97	61	A	a	a
98	62	B	b	b
99	63	C	c	c
100	64	D	d	d
101	65	E	e	e
102	66	F	f	f
103	67	G	g	g
104	68	H	h	h
105	69	I	i	i

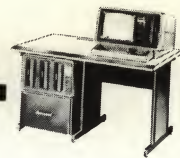


Model II TRSDOS

Code		Character		
Dec.	Hex.	Key-board	Video Display	
			Scroll mode	Graphics mode
106	6A	J	j	j
107	6B	K	k	k
108	6C	L	l	l
109	6D	M	m	m
110	6E	N	n	n
111	6F	O	o	o
112	70	P	p	p
113	71	Q	q	q
114	72	R	r	r
115	73	S	s	s
116	74	T	t	t
117	75	U	u	u
118	76	V	v	v
119	77	W	w	w
120	78	X	x	x
121	79	Y	y	y
122	7A	Z	z	z
123	7B	}	}	}
124	7C	CTRL 0	:	:
125	7D	}	}	}
126	7E	CTRL 6	~	~
127	7F			±
128	80	*		
:	:			
:	:			
248	F8			
249	F9			Sets white-on-black mode
250	FA			Sets black-on-white mode
251	FB			Homes cursor
252	FC			Moves cursor back
253	FD			Moves cursor forward
254	FE			Moves cursor up
255	FF			Moves cursor down

* Codes 128-248 cannot be input from the keyboard or output to the display. When reading the display, a **value** greater than 127 indicates a reverse character corresponding to **value mod 128**.





Decimal–Hexadecimal Conversion

DEC HEX

0 00
1 01
2 02
3 03
4 04

5 05
6 06
7 07
8 08
9 09

10 0A
11 0B
12 0C
13 0D
14 0E

15 0F
16 10
17 11
18 12
19 13

20 14
21 15
22 16
23 17
24 18

25 19
26 1A
27 1B
28 1C
29 1D

30 1E
31 1F
32 20
33 21
34 22

DEC HEX

35 23
36 24
37 25
38 26
39 27

40 28
41 29
42 2A
43 2B
44 2C

45 2D
46 2E
47 2F
48 30
49 31

50 32
51 33
52 34
53 35
54 36

55 37
56 38
57 39
58 3A
59 3B

60 3C
61 3D
62 3E
63 3F
64 40

65 41
66 42
67 43
68 44
69 45

DEC HEX

70 46
71 47
72 48
73 49
74 4A

75 4B
76 4C
77 4D
78 4E
79 4F

80 50
81 51
82 52
83 53
84 54

85 55
86 56
87 57
88 58
89 59

90 5A
91 5B
92 5C
93 5D
94 5E

95 5F
96 60
97 61
98 62
99 63

100 64
101 65
102 66
103 67
104 68

DEC HEX

105 69
106 6A
107 6B
108 6C
109 6D

110 6E
111 6F
112 70
113 71
114 72

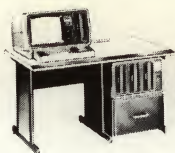
115 73
116 74
117 75
118 76
119 77

120 78
121 79
122 7A
123 7B
124 7C

125 7D
126 7E
127 7F
128 80
129 81

130 82
131 83
132 84
133 85
134 86

135 87
136 88
137 89
138 8A
139 8B



MODEL II TRSDOS

Decimal-Hexadecimal Conversion, continued

DEC	HEX	DEC	HEX	DEC	HEX
140	8C	180	B4	220	DC
141	8D	181	B5	221	DD
142	8E	182	B6	222	DE
143	8F	183	B7	223	DF
144	90	184	B8	224	E0
145	91	185	B9	225	E1
146	92	186	BA	226	E2
147	93	187	BB	227	E3
148	94	188	BC	228	E4
149	95	189	BD	229	E5
150	96	190	BE	230	E6
151	97	191	BF	231	E7
152	98	192	C0	232	E8
153	99	193	C1	233	E9
154	9A	194	C2	234	EA
155	9B	195	C3	235	EB
156	9C	196	C4	236	EC
157	9D	197	C5	237	ED
158	9E	198	C6	238	EE
159	9F	199	C7	239	EF
160	A0	200	C8	240	F0
161	A1	201	C9	241	F1
162	A2	202	CA	242	F2
163	A3	203	CB	243	F3
164	A4	204	CC	244	F4
165	A5	205	CD	245	F5
166	A6	206	CE	246	F6
167	A7	207	CF	247	F7
168	A8	208	D0	248	F8
169	A9	209	D1	249	F9
170	AA	210	D2	250	FA
171	AB	211	D3	251	FB
172	AC	212	D4	252	FC
173	AD	213	D5	253	FD
174	AE	214	D6	254	FE
175	AF	215	D7	255	FF
176	B0	216	D8		
177	B1	217	D9		
178	B2	218	DA		
179	B3	219	DB		

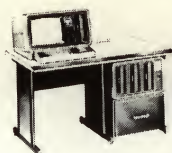
Section 6

Index



Subject	Page
Access word	1/14, 2/9
AGAIN	2/6
APPEND	2/4, 2/7
ARCV	4/78, 4/79
ATTRIB	1/4, 1/14, 2/9, 2/55
ATX	4/79
AUTO	2/4, 2/11
BACKUP	1/5, 2/6, 3/3, 3/4-3/5, 4/19
BASCOM	3/9
BASIC	1/5, 2/11, 2/25, 2/38, 2/51, 2/56
BINDEC	4/64
BINHEX	4/67
BRCV	4/80, 4/81
BTX	4/80, 4/81
BUILD	1/10, 2/4, 2/13, 2/38, 2/39
Caps mode	1/7
Channel A	2/56-2/57, 4/76-4/79
Channel B	2/25, 2/56-2/57, 4/76-4/77, 4/80-4/81
CLEAR	2/16
CLOCK	2/17
CLOSE	4/58
CLS	2/18
Command mode	1/7, 2/3
Command syntax	1/9
Computational supervisor calls	4/61
COMSUB	3/9
Control code	1/8, 4/35, 4/41
Control key	1/8
COPY	1/5, 1/14, 2/4, 2/6, 2/15, 2/19-2/20
CREATE	1/9, 2/4, 2/21-2/23, 4/4
CURSOR	4/43
Data Control Block	4/53
DATE	2/4, 2/24, 2/58, 4/68
DATM	3/12
DB-25 connector	4/77
DEBUG	2/25-2/34, 2/38, 2/40, 4/83
Decimal to hexadecimal conversion table	5/11-5/12
DELAY	4/20, 4/62
DIR	1/9, 2/4, 2/11, 2/35-2/37
Direct file access	2/22, 4/7-4/8, 4/52
DIRRD	4/52
DIRWR	4/60
Diskette names	1/14
Diskette organization	4/3
DISKID	4/19
DO	1/10, 2/13, 2/38-2/39
DOSCMD	4/22
Drive specification	1/13
DUMP	2/4, 2/40, 4/83
Dynamically allocated file	2/23, 4/4
Entering a Command	1/9, 2/4
ERRMSG	4/25
ERROR	2/4, 2/41, 4/24

Subject	Page
Error messages	2/6, 2/41, 4/10
EXDATM	3/12
Examples of Syntax Forms	1/11
File access supervisor calls	4/49
File allocation, methods of	4/4
File names	1/12
File specification	1/12
File type	2/22
Fixed-length record files	2/22, 4/5-4/7, 4/50, 4/54-4/55
FORMS	2/42-2/43
FORMAT	1/5, 3/3, 3/6-3/7, 4/19
FREE	2/44-2/45
Function code	4/9
Granule	2/22, 2/44, 4/3-4/4
Graphics control codes	4/38
Graphics Mode	4/32, 4/38, 4/40
HERZ50	3/27
I	2/46
INITIO	4/16
Input/output drivers	4/16, 4/28
Interrupts	4/84
JP2DOS	4/21
KBCHAR	4/29
KBINIT	4/28
KBLINE	4/30, 4/41
Keyboard code map	5/7
Keyboard supervisor calls	4/27
KILL	2/4, 2/47, 4/57
LIB	2/48
Library Commands	1/3, 2/3
Line printer supervisor calls	4/45
LIST	1/4, 1/9, 2/4, 2/49-2/50
List Address Block	4/70-4/72
LOAD	2/51
Loading TRSDOS	1/6
LOCATE	4/50
Memory Requirements	1/5
Modulo	4/39
MPYDIV	4/66
Name-extensions	1/12
No-file commands	1/10, 1/11
Notation	1/4
One-file commands	1/10, 1/11
OPEN	4/53
Parameter error	1/8
PARSER	4/69-4/73
Passwords	1/13, 2/9, 2/54



MODEL II TRSDOS

Subject	Page
PATCH	3/28
PAUSE	2/4, 2/38, 2/52
PRCHAR	4/47
Pre-Allocated Files	4/4
PRINIT	4/46
PRLINE	4/48
Programmed entry conditions	4/83
Programmed interrupts	4/83-4/84
Programming with TRSDOS	4/83
PROT	1/14, 2/54
PURGE	2/4, 2/53
RANDOM	4/63
READNX	4/51
Record	2/22
Record length	2/22, 4/4-4/6
RENAME	2/20, 2/55
Repeat key	1/8
RETCMD	4/23
Reverse mode characters	4/40
RST 8 instruction	4/9, 4/17
RS232C	4/76
SCROLL	4/33, 4/44
Scroll mode	4/30, 4/33, 4/37, 4/41
Sectors	4/3-4/6
Sequential file access	4/7-4/8
Serial communications supervisor calls ...	4/75
Serial input mode	2/32
Serial interface	2/25, 2/32
SETBRK	2/38, 4/18, 4/29, 4/84
SETCOM	2/56-2/57, 4/77
SETUSR	4/17
Spanning	4/5
STCMP	4/65
STSCAN	4/74
Supervisor calls	4/9
Syntax Forms	1/10
SYS	2/5, 2/11, 2/35
SYSTEM	2/38
System control supervisor calls	4/15
System diskette	1/6
System routines	1/3
Tab positions	1/8
TERMINAL	3/33
TIME	2/4, 2/6, 2/58
TIMER	4/20, 4/83-4/84
Tracks	4/3
TRSDOS character codes	5/3-5/6
TRSDOS READY prompt	1/9, 2/3
Two-file commands	1/10, 1/11
Update word	1/14, 2/9, 2/14
Upload function	2/25
Upper and lower case letters	1/7
User programs, location of	1/5
User vector	4/17
Using the Keyboard	1/7
Utility programs	1/3, 3/3

Subject	Page
Variable-length record files	2/22, 4/5-4/8, 4/52, 4/55, 4/60
VDCHAR	4/35, 4/37
VDGRAF	4/38
VDINIT	4/34
VDLINE	4/37, 4/41
VDREAD	4/40
VERIFY	2/4, 2/59
Versions of TRSDOS	1/5
Video Display Supervisor Calls	4/31
VIDKEY	4/41
WRITNX	4/59
XFERSYS	3/55

